









25

2018

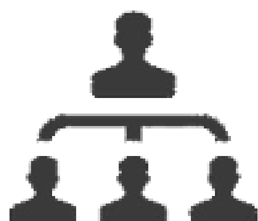
10 Años Aportando al Conocimiento

2018 - Bolvia



http://revista.atixlibre.org
Twitter: @atixlibre
Facebook: facebook.com/Atix.Libre





DIRECCION GENERAL
Esteban Saavedra Lopez



DIAGRAMACION
Jenny Saavedra Lopez
Esteban Saavedra Lopez



REVISION
Jenny Saavedra Lopez



CONTACTO
info@atixlibre.org
http://revista.atixlibre.org







AtixLibre



EL QUE LO INTENTA

EL QUE LO SABE

EL QUE LO PUEDE

EL QUE LO LOGRA

Si bien el tema de tecnologías libres tiene varios años entre nosotros, los avances tecnológicos en este área se dan día a día, creando nuevas herramientas y cambiando los modelos de negocio tradicionales de como se venia manejando el desarrollo de software y hardware; de forma general creando nuevos paradigmas y metodologías que aceleran la evolución tecnológica.

Muchos hemos visto y conocido de cerca la importancia de este cambio, pero aún son pocas las personas o instituciones que se han subido a este barco y aún mantienen conceptos y tecnologías del siglo pasado, aspecto que les empieza a cobrar factura sobre todo por que la evolución tecnológica va creando brechas entre quienes o no se sumergen en ésta.

Muchos aún piensan que las tecnologías libres, están solamente orientada a las personas del ámbito informático, sin considerar, que es un tema globalizado, que ha revolucionado y transformado diversos mercados, un aspecto a destacar es el ámbito de la investigación y la educación, que hoy por hoy ha encontrado en el uso de las tecnologías libres su piedra fundamental para su desarrollo.

En AtixLibre nos sentimos orgullosos, que en cada uno de nuestros números, aportemos con nuevos y mayores conocimientos, principalmente destinados aquellas personas que poseen esa sed de alimentar su conocimiento hacia las tecnologías libres.

Sean bienvenidos a nuestra edición número 25.



Esteban Saavedra L.

Presidente Fundación AtixLibre

Contenido

Número 25 - Agosto 2018



MkDocs
Documentando Nuestros Proyectos

Casi desde siempre documentar los proyectos de desarrollo se convirtieron en el punto débil de cualquier proyecto de desarrollo, ya que representaba desgastarse en procesos tediosos, más aún cuando no se contaba con entornos acordes a este trabajo, pero hoy en día las cosas han cambiado, ya que existen numerosas herramientas que nos permiten implementar mecanismos ágiles y automatizados para documentar toda clase de proyectos.

Una corriente que ha tomado gran fuerza son las herramientas basadas en estándares como son los textos restructurados (rst) y los archivos markdown (md).

Introducción

Este articulo nace por la motivación del Ing Esteban Saavedra, quién fue tutor de mi proyecto de tesis, y me invitó a compartir mi experiencia en el manejo de una herramienta de código abierto orientada a la gestión de documentación de proyectos, adicionalmente fue quién me enseño lo importante de contar con una herramienta que permita documentar todas y cada una de las etapas que demanda hacer una tesis, haciendo particular énfasis en que la herramienta de documentación utilizada sea fácil de comprender, manejar y ligera al momento de transportar y desplegar en sitios públicos o privados.

Este tipo de herramientas no solo sirven para documentar proyectos de software, sino que son perfectamente adaptables a cualquier tipo de proyectos ya sean estos de desarrollo o de investigación, ya que sobre todo que utilizan una nomenclatura fácil de comprender, recordar y pueden ser exportados a distintos tipos de formato para su despliegue o visualización final.

Documentar un proyecto

La herramienta que se utilice para la documentación de un proyecto debe ser accesible, fácil de leer, de estructurar y de editar. Adicionalmente debe poder permitir mantener el historial de cambios, para esto debe ir asociada a un control de versiones y debe tener la capacidad de exportar a distintos formatos de visualización (html, pdf, ebook, etc)

MarkDown

Markdown es un lenguaje de marcas ligero que trata de conseguir la máxima legibilidad y “publicabilidad” tanto en sus forma de entrada como de salida.

Beneficios de MakDown

- Herramienta de redacción clara y fácil de usar
- Las marcas nos permiten diseñar la estructura del documento
- Trabajamos en distintos niveles, diferenciamos el material generado considerando el

punto de vista del autor y de la visión del lector en el producto final.

- Nos permite separar las ideas en líneas cortas, (si no generamos líneas en blanco, se mantienen los párrafos).
- Es ligero, fácil de leer, fácil de escribir, compatible con multitud de plataformas (portátiles, teléfonos móviles, tablets, etc).
- El peso de los archivos es ínfimo en comparación con los archivos generados por los procesadores de texto, así mismo al tener un formato de texto plano también permite que se pueda utilizar un sistema de control de versiones y de esta forma poder contar con todos los beneficios que denota el uso de estos.

MkDocs

Es una herramienta escrita en python, que permite generar sitios estáticos, orientado a documentar proyectos y desplegarlos mediante un servidor web.

MkDocs hace uso del lenguaje de marcado Markdown, para la elaboración del contenido de la documentación.

MkDocs

Project documentation with Markdown

Características

- Sencillo y ligero en su manejo
- Los archivos fuente son escritos haciendo uso de la nomenclatura Markdown
- Es posible utilizar extensiones, para ampliar sus capacidades, por ejemplo, para resaltar en colores código, uso de bloques personalizados, etc.
- Precisa de sólo un simple archivo de configuración YAML
- Permite establecer temas en su plataforma de documentación
- Permite crear su propio buscador de texto
- Esta provisto de un servidor web, que permite mostrar todos los cambios realizados de forma inmediata
- Permite trabajar con herramientas como Read The Docs y PyPI
- Capacidad de interactuar con un sistema de control de versiones, lo que permite:
    - Poder otorgar permisos de edición a quien lo necesita.
    - Poder seguir los cambios realizados.

- Tener copias de la documentación.

Gestión de la plataforma de documentación

El objetivo de este artículo, no es detallar el uso del lenguaje marcado markdown, sino la implementación de una plataforma de documentación basada en MkDocs.

Instalación

Debian/Ubuntu

```
$ apt install python-pip
$ pip install mkdocs
```

Versión instalada

```
$ mkdocs --version
mkdocs, version 1.0 from /usr/local/lib/python2.7/dist-packages/mkdocs (Python 2.7)
```

Crear un proyecto

```
$ mkdocs new proyecto
INFO    -  Creating project directory: proyecto
INFO    -  Writing config file: proyecto/mkdocs.yml
INFO    -  Writing initial docs: proyecto/docs/index.md
```

Estructura de un proyecto

```
├── docs
│   └── index.md
└── mkdocs.yml
```

Donde:

- **docs**: es la carpeta donde se almacenarán todos los archivos y subcarpetas de documentación del proyecto.
- **mkdocs.yml**: es el archivo de configuración del proyecto, donde se pueden configurar aspectos como: metadatos, temas, complementos, plugins, etc.

Servidor de despliegue

Mkdocs cuenta con un servidor web que nos permite desplegar nuestro proyecto de documentación.

```
$ mkdocs serve
INFO    -  Building documentation...
INFO    -  Cleaning site directory
[I 180812 11:43:30 server:292] Serving on http://127.0.0.1:8000
[I 180812 11:43:30 handlers:59] Start watching changes
[I 180812 11:43:30 handlers:61] Start detecting changes
```

Documentación temática

La mejor forma de organizar la documentación, dentro de MkDocs es organizándola por carpetas, subcarpetas y páginas de contenidos, considerando que la interacción entre los distintos contenidos son hiperenlaces entre páginas, contenidos, imágenes, etc.

Partes de la documentación

El archivo **`mkdocs.yml`** contiene todas las partes que permiten definir las características propias del proyecto, como ser:

- Nombre del proyecto
- Descripción del proyecto
- URL del repositorio
- URL del proyecto
- Metadatos del proyecto
- Personalización del proyecto
- Navegabilidad

Nombre del proyecto

mkdocs.yml

```
site_name: Plataforma eComerce
```

Navegabilidad del proyecto

Permite establecer el nombre de las páginas y los archivos o sitios asociados a las mismas (locales o remotos), que determinan la navegabilidad del proyecto.

```
nav:
    - 'Introduccion': 'index.md'
    - 'Guia de Usuario': 'guia.md'
    - 'Acerca de': 'about.md'
    - Bug Tracker: https://misitio.org/proyecto/
```

También es posible hacer uso de rutas relativas.

```
site_url: https://misitio.org/guias/

nav:
    - Inicio: ../
    - Guia_de_usuario: guia.md
    - Bug Tracker: /bugs/
```

Organización por carpetas y archivos específicos

Por carpetas

En este caso se asume que dentro de cada carpeta existe un archivo **`index.md`**

```
docs/
    index.md
    guia_de_usuario/instalacion/
    guia_de_usuario/configuracion/
    licencia/
```

Por archivos específicos

```
docs/
    index.md
    guia_de_usuario/instalacion.md
    guia_de_usuario/configuracion.md
    licencia.md
```

Archivos de documentación

Los archivos de documentación, para establecer su contenidos deben hacer uso de las marcas provistas por markdown y por ende todo el potencial y sencillez que representa este, aquí algunos ejemplos de los mismos:

3. Resaltado de codigo fuente

Configuración apache

```
<VirtualHost *:80>
DocumentRoot /www/example1
ServerName www.example1.com

# Other directives here

</VirtualHost>
```

Codigo Python, resaltando lineas especificas

```
""" Bubble sort """
def bubble_sort(items):
    for i in range(len(items)):
        for j in range(len(items) - 1 - i):
            if items[j] > items[j + 1]:
                items[j], items[j + 1] = items[j + 1], items[j]
```

2. Manejo de formulas

De forma separada

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Dentro de un texto Lorem ipsum dolor sit amet: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

Publicar el proyecto

Hasta este momento nuestra plataforma de documentación es totalmente funcional localmente, pero si surge la necesidad de poder publicarlo en un sitio público o en un intranet, se recomienda compilar el sitio, esto representa transformarlo en formato html.

```
$ mkdocs build
```

Una vez compilado el proyecto, el directorio contendrá una serie de archivos (html, ccs, js, json, svg, etc) los cuales pueden ser alojados en un servidor público o intranet, permitiendo desplegar un sitio totalmente funcional.

El contenido del directorio será:

```
├── docs
│   └── index.md
├── mkdocs.yml
└── site
    ├── 404.html
    ├── css
    │   ├── base.css
    │   ├── bootstrap-custom.min.css
    │   └── font-awesome.min.css
    ├── fonts
    │   ├── fontawesome-webfont.eot
    │   ├── fontawesome-webfont.svg
    │   ├── fontawesome-webfont.ttf
    │   ├── fontawesome-webfont.woff
    │   ├── fontawesome-webfont.woff2
    │   ├── glyphicons-halflings-regular.eot
    │   ├── glyphicons-halflings-regular.svg
    │   ├── glyphicons-halflings-regular.ttf
    │   ├── glyphicons-halflings-regular.woff
    │   └── glyphicons-halflings-regular.woff2
    ├── img
    │   ├── favicon.ico
    │   └── grid.png
    ├── index.html
    ├── js
    │   ├── base.js
    │   ├── bootstrap-3.0.3.min.js
    │   └── jquery-1.10.2.min.js
    ├── search
    │   ├── lunr.js
    │   ├── main.js
    │   ├── search_index.json
    │   └── worker.js
    ├── sitemap.xml
    └── sitemap.xml.gz
```

Despliegue en GitHub

Desplegar nuestro proyecto de documentación en GitHub, para la gestión del mismo, es tan sencillo como hacer un click en el icono de GitHub presente en nuestra plataforma o desde la consola realizar los siguientes pasos:

Inicializar el control, de versiones

```
$ git init
```

Añadir nuestro sitio en GitHub

```
$ git remote add origin https://github.com/atixlibre/proyecto.git
```

Añadir todos los archivos

```
$ git add -A
```

Realizar el commit y documentarlo

```
$ git commit -a -m 'inicio del proyecto de documentación'
```

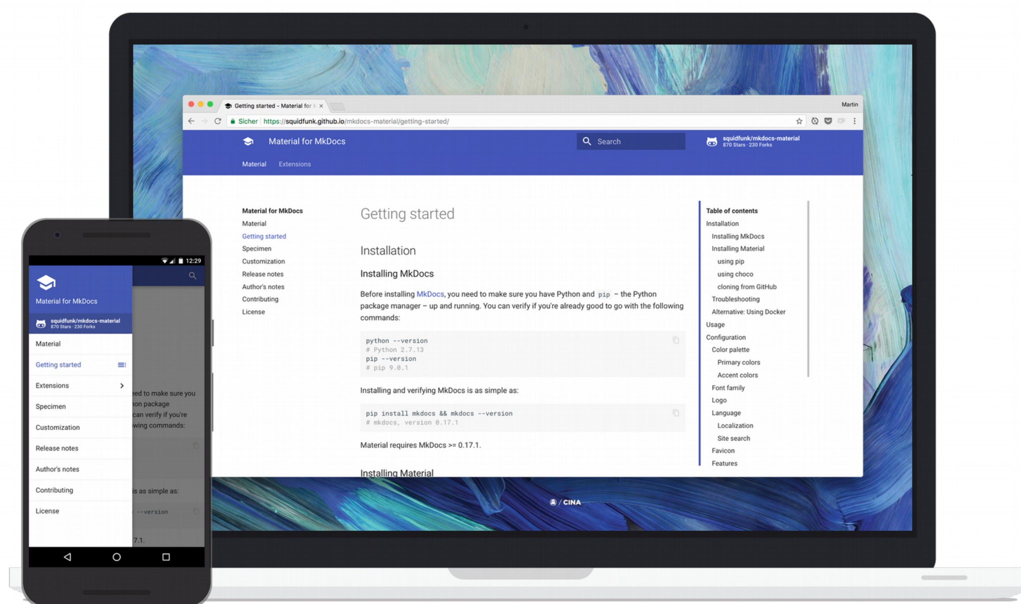
Transferir los archivos a GitHub

```
git push --set-upstream origin master
```

Personalizando el entorno

Personalizar nuestro entorno de documentación, representa entre otras cosas:

- Cambiar el tema o hacer modificaciones sobre el mismo, lo que nos permite:
    - Disponer de un diseño responsive, que permite un despliegue fluido en toda la variedad de dispositivos
    - Facilidad de personalizar, colores, fuentes, hojas de estilo, sobreescritura de bloques dentro las plantillas, etc
    - Nuevos diseños y manejo de hotkeys para acceder a búsquedas y navegación.
- Instalación de plugins y complementos para añadir nuevas funcionalidades o perfeccionar la visualización de los documentos



Manejo de temas

Mkdocs trae un tema por defecto, pero dispone de la posibilidad de poder añadir, personalizar o crear nuevos temas previa instalación de los mismos.

Algunos ejemplos de temas son mostrados a continuación:

Proyecto eCommerce - Mozilla Firefox

Proyecto eComerce

127.0.0.1:8000

Buscar

Inicio

Search

GitHub

Proyecto eComerce

Inicio

Capitulos

Capitulo 1

Capitulo 2

Capitulo 3

Capitulo 4

Demostración

Codigo Fuente

Acerca de

Contacto

1.1. Historia del arte

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

1.1.1. Comienzos

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

2. Manejo de formulas

De forma separada

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Dentro de un texto Lorem ipsum dolor sit amet: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

3. Resaltado de codigo fuente

Table of contents

1.1. Historia del arte

1.1.1. Comienzos

Proyecto eComerce - Mozilla Firefox

Proyecto eComerce

127.0.0.1:8000

Buscar

Proyecto eComerce | Inicio | Capitulos | Acerca de | Contacto | Search | Edit on GitHub

1.Introduccion

1.1. Historia del arte

2. Manejo de formulas

3. Resaltado de codigo fuente

Capitulo 1

Capitulo 2

Capitulo 3

Capitulo 4

Demostración

Codigo Fuente

t egestas elementum at neque.

del arte

et consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra t egestas elementum at neque.

os

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

2. Manejo de formulas

De forma separada

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Dentro de un texto Lorem ipsum dolor sit amet: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

3. Resaltado de codigo fuente

Configuración apache

127.0.0.1:8000/capitulo02.md

Inicio - Proyecto eComerce - Mozilla Firefox

Inicio - Proyecto eCo... ×

127.0.0.1:8000

Buscar

Proyecto eComerce

Search docs

Inicio

1.Introduccion

1.1. Historia del arte

2. Manejo de formulas

3. Resaltado de codigo fuente

Capitulos

Capitulo 1

Capitulo 2

Capitulo 3

Capitulo 4

Demostración

Codigo Fuente

Acerca de

Contacto

GitHub

1.1. Historia del arte

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

1.1.1. Comienzos

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

2. Manejo de formulas

De forma separada

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Dentro de un texto Lorem ipsum dolor sit amet: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

3. Resaltado de codigo fuente

Configuración apache

Proyecto eComerce - Mozilla Firefox

Proyecto eComerce ×

127.0.0.1:8000

Buscar

Proyecto eComerce

Search

GitHub

Inicio

Capitulos

Capitulo 1

Capitulo 2

Capitulo 3

Capitulo 4

Demostración

Codigo Fuente

Acerca de

Contacto

1.Introduccion

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

1.1. Historia del arte

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

1.1.1. Comienzos

Lorem ipsum dolor sit amet consectetur adipiscing elit cras, malesuada arcu libero pharetra etiam viverra cubilia, nunc fusce feugiat egestas elementum at neque.

2. Manejo de formulas

De forma separada

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

Dentro de un texto Lorem ipsum dolor sit amet: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$



Instalar temas

```
$ pip install mkdocs-material
```

Añadir un tema

mkdocs.yml

```
theme:
  name: material
```

Un listado y ejemplos de algunos temas disponibles se puede encontrar en **`https://github.com/mkdocs/mkdocs/wiki/MkDocs-Themes`**

Exportar la documentación

Como mencionamos, una de las características imprescindibles de una plataforma de documentación es la capacidad de poder exportar su contenido a distintos formatos; en el caso particular de MkDocs podemos interactuar con la herramienta **pandoc**, que es una herramienta sumamente potente y fácil de usar, así por ejemplo para exportar un archivo **`.md`** a un archivo **`.pdf`** solo es necesario:

```
$ pandoc manual.md -o manual.pdf
```

Conclusiones

- La documentación es un elemento sumamente importante e imprescindible dentro un proyecto de desarrollo o investigación, de tal manera que este debe ir de forma paralela al trabajo que se realiza y no dejarlo como última actividad ya que descuidaríamos varios detalles.
- Para que el documentar cualquier proyecto no represente una tarea tediosa o aburrida, debemos emplear herramientas que sean fáciles de manejar, comprender y transportar.

Referencias

[1] http://www.mkdocs.org



Ninoska Gutierrez G.
Ingeniero de Sistemas
ninoska.carol.gutierrez@gmail.com
BOLIVIA



Vagrant

Automatizando Entornos Virtuales

En un contexto tecnológico donde el común denominador es el término virtualización, han surgido numerosas herramientas que permiten de una u otra forma gestionar este tipo de entornos, dándonos la facilidad de poder preparar, configurar, desplegar y compartir entornos de trabajo ya sea para desarrollos, demostraciones o entornos de producción.

Estas herramientas permiten a las unidades de desarrollo, QA e infraestructura optimizar su trabajo y reducir considerablemente el tiempo y la carga de trabajo en la ejecución de estos entornos.

Introducción

En el anterior número de la revisa, hicimos una descripción detalladas de las características y la gestión de funcionalidades básicas de Vagrant.

En este número, veremos como gestionar Boxes (máquinas virtuales) de forma grupal y la interacción entre las mismas, así mismo veremos como crear nuestra propia Box en base a alguna distribución personalizada.

Gestión de Boxes

A continuación mostramos algunos de los comandos más útiles para la gestión de Box desde Vagrant.

Listado de las Box añadidas

```
$ vagrant box list
```

Verificar actualizaciones

Muchas de las box añadidas, puede que hayan sido actualizadas por sus creadores, una forma de ver cuales están desactualizadas es:

```
$ vagrant box outdated
```

Actualizar una Box

```
$ vagrant box update
```

Remover una Box

Dado que alguna vez dejemos de usar alguna Box podemos removerla

```
$ vagrant box remove debian/stretch64
```

En caso de querer remover una box de cierto provider.

```
$ vagrant box remove debian/stretch64
--provider virtualbox
```

Estado global de las Boxs

```
$ vagrant global-status
id       name     provider   state
directory
-----------------------------------------
-------------------------------
646342d  default virtualbox poweroff
/home/atixlibre/vagrant/proyecto02

3ffea73  default virtualbox poweroff
/home/atixlibre/vagrant/proyecto03
```

Detener una Box en específico

```
$ vagrant suspend ID
$ vagrant suspend 646342d
```

Gestión de múltiples máquinas virtuales

El desarrollo de aplicaciones ha tomado el cause de un desarrollo por partes, siguiendo la línea del desarrollo de microservicios, en ese entendido las aplicaciones modernas están siendo desarrolladas utilizando múltiples y distintas partes, en algún momento denominados servicios, un claro ejemplo es disponer de forma separada de un servidor de aplicaciones, un servidor de base de datos, un broker de mensajería, un manejador de datos en cache, etc; los cuales pueden ser integrados en distintos proyectos como partes de un todo.



En estos casos es imperioso disponer de un conjunto de máquinas, las cuales puedan alojar cada componente (servicio) en una máquina distinta; para esto Vagrant surge como una herramienta acorde a estas necesidades.

Definición de múltiples máquinas

La sintaxis utilizada para la definición de múltiples máquinas es la siguiente:

```
config.vm.define :nombre_de_la_vm do |
nombre_de_la_vm|
    #configuracion especifica de la
     maquina virtual
end
```

Para un proyecto donde se requiera dos máquinas virtuales la configuración llegaría a ser:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
# Vagrantfile API/syntax version. Don't
touch unless you know what you're doing!
Vagrant.configure("2") do |config|

 config.vm.define :servidor01 do
 |servidor01|
   servidor01.vm.box = "centos/7"
   servidor01.vm.hostname = "servidor01"
 end

 config.vm.define :servidor02 do
 |servidor02|
   servidor02.vm.box = "debian/stretch64"
   servidor02.vm.hostname = "servidor02"
 end
end
```


Conexión de múltiples máquinas

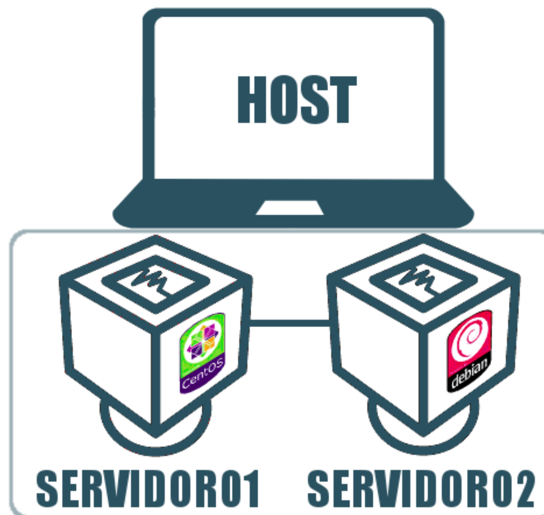
Por defecto cuando ejecutamos un proyecto de múltiples máquinas, Vagrant automáticamente mapea los diferentes puertos desde nuestro host a los puertos SSH de las máquinas virtuales.

Conexión individual

```
$ vagrant ssh servidor01
$ vagrant ssh servidor02
```

Conexión de red de múltiples máquinas

Para que las múltiples máquinas que componen nuestro proyecto, puedan comunicarse entre sí, es necesario establecer una red privada entre ellas.




```
# -*- mode: ruby -*-
# vi: set ft=ruby :
# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
Vagrant.configure("2") do |config|
 config.vm.define :servidor01 do |servidor01|
    servidor01.vm.box = "centos/7"
    servidor01.vm.hostname = "servidor01"
    servidor01.vm.network "private_network", ip: "10.11.1.100"
 end
 config.vm.define :servidor02 do |servidor02|
    servidor02.vm.box = "debian/stretch64"
    servidor02.vm.hostname = "servidor02"
    servidor02.vm.network "private_network", ip: "10.11.1.101"
 end
end
```

Test de conexión

Los siguientes pasos nos ayudan a probar la conectividad entre las múltiples máquinas que componen nuestro proyecto:

Iniciar nuestro proyecto

```
$ vagrant up
```

Conexión al servidor01

```
$ vagrant ssh servidor01
```

Test desde el servidor01

```
$ ping 10.11.1.101
```

Eliminación de un proyecto múltiple

```
$ vagrant destroy nombre_proyecto
```

Provisión de un proyecto múltiple

Vagran nos da la posibilidad de hacer uso de cualquier método de provisión de un proyecto de múltiples máquinas, teniendo en cuenta que cada máquina componente puede tener una aprovisionamiento distinto e independiente.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
# Vagrantfile API/syntax version. Don't touch unless you know what you're doing!
Vagrant.configure("2") do |config|
 config.vm.define :servidor01 do |servidor01|
    servidor01.vm.box = "centos/7"
    servidor01.vm.hostname = "servidor01"
    servidor01.vm.network "private_network", ip: "10.11.1.100"
    servidor01.vm.provision :shell, path: "provision01.sh"
 end
 config.vm.define :servidor02 do |servidor02|
    servidor02.vm.box = "debian/stretch64"
    servidor02.vm.hostname = "servidor02"
    servidor02.vm.network "private_network", ip: "10.11.1.101"
    servidor02.vm.provision :shell, path: "provision02.sh"
 end
end
```

Control de un proyecto múltiple

Recarga el proyecto múltiple

```
$ vagrant reload
```

Recarga de una máquina en específica

```
$ vagrant reload servidor01
```

Recarga de varias máquinas en específico

```
$ vagrant reload servidor01 servidor02
```

Estado general

```
$ vagrant status
```

Estado de una Box en específico

```
$ vagrant status servidor01
```

Entorno de demostración

En el siguiente ejemplo, configuraremos un entorno de múltiples máquinas, bajo las siguientes características:



Un servidor (Debian 9.4) de base de datos, que contenga mysql y permita la conexión remota; este servidor tendrá instalado el servidor de base de datos MySQL.



Un máquina cliente (Centos 7), que contenga el cliente de MariaDB (MariaDB-client) y pueda conectarse al servidor de base de datos de forma remota.

En ambos casos se dispone de un aprovisionamiento de software, en el caso del servidor se realiza mediante un script de aprovisionamiento y en el caso del cliente se realiza mediante una aprovisión en línea. Esta aprovisión permitirá que la primera vez que se ejecute vagrant, configure las box e instale y realice la configuración necesaria de los paquetes en cada una de ellas.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
 config.vm.define :cliente do |cliente|
    cliente.vm.box = "centos/7"
    cliente.vm.hostname = "cliente"
    cliente.vm.provision :shell, inline: "cat
/vagrant/MariaDB.repo>/etc/yum.repos.d/MariaDB.repo"
    cliente.vm.provision :shell, inline: "yum install MariaDB-client -y"
    cliente.vm.network "private_network", ip: "192.168.33.10"
    cliente.vm.provider :virtualbox do |cli|
       cli.customize ["modifyvm", :id, "--memory", 512]
       cli.customize ["modifyvm", :id, "--name", "cliente"]
    end
 end
 config.vm.define "servidor" do |servidor|
    servidor.vm.box = "debian/stretch64"
    servidor.vm.hostname = "servidor"
    servidor.vm.provision :shell, path: "svr_provision.sh"
    servidor.vm.network "private_network", ip: "192.168.33.11"
    servidor.vm.provider :virtualbox do |svr|
       svr.customize ["modifyvm", :id, "--memory", 512]
       svr.customize ["modifyvm", :id, "--name", "servidor"]
    end
 end
end
```

donde:

```
srv_provision.sh
export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get install -y mysql-server
sed -i -e 's/127.0.0.1/0.0.0.0/' /etc/mysql/mariadb.conf.d/50-server.cnf
systemctl restart mysql
mysql -uroot mysql <<< "GRANT ALL PRIVILEGES ON *.* TO atix@'%' IDENTIFIED BY 'clave';
FLUSH PRIVILEGES;"
```

Ejecución múltiple

```
$ vagrant up
Bringing machine 'cliente' up with 'virtualbox' provider...
Bringing machine 'servidor' up with 'virtualbox' provider...
==> cliente: Checking if box 'centos/7' is up to date...
==> cliente: Clearing any previously set forwarded ports...
==> cliente: Clearing any previously set network interfaces...
==> cliente: Preparing network interfaces based on configuration...
    cliente: Adapter 1: nat
    cliente: Adapter 2: hostonly
==> cliente: Forwarding ports...
    cliente: 22 (guest) => 2222 (host) (adapter 1)
==> cliente: Running 'pre-boot' VM customizations...
==> cliente: Booting VM...
==> cliente: Waiting for machine to boot. This may take a few minutes...
    cliente: SSH address: 127.0.0.1:2222
    cliente: SSH username: vagrant
    cliente: SSH auth method: private key
==> cliente: Machine booted and ready!
==> cliente: Checking for guest additions in VM...
    cliente: No guest additions were detected on the base box for this VM! Guest
    cliente: additions are required for forwarded ports, shared folders, host only
    cliente: networking, and more. If SSH fails on this machine, please install
    cliente: the guest additions and repackage the box to continue.
    cliente:
    cliente: This is not an error message; everything may continue to work properly,
    cliente: in which case you may ignore this message.
==> cliente: Setting hostname...
==> cliente: Configuring and enabling network interfaces...
    cliente: SSH address: 127.0.0.1:2222
    cliente: SSH username: vagrant
    cliente: SSH auth method: private key
==> cliente: Rsyncing folder: /home/atixlibre/vagrant/multiple/ => /vagrant
==> cliente: Machine already provisioned. Run `vagrant provision` or use the `--
provision`
==> cliente: flag to force provisioning. Provisioners marked to run always will still
run.
==> servidor: Checking if box 'debian/stretch64' is up to date...
==> servidor: Clearing any previously set forwarded ports...
==> servidor: Fixed port collision for 22 => 2222. Now on port 2200.
==> servidor: Clearing any previously set network interfaces...
==> servidor: Preparing network interfaces based on configuration...
    servidor: Adapter 1: nat
    servidor: Adapter 2: hostonly
==> servidor: Forwarding ports...
    servidor: 22 (guest) => 2200 (host) (adapter 1)
==> servidor: Running 'pre-boot' VM customizations...
==> servidor: Booting VM...
==> servidor: Waiting for machine to boot. This may take a few minutes...
    servidor: SSH address: 127.0.0.1:2200
    servidor: SSH username: vagrant
    servidor: SSH auth method: private key
```

```
==> servidor: Machine booted and ready!
==> servidor: Checking for guest additions in VM...
    servidor: No guest additions were detected on the base box for this VM! Guest
    servidor: additions are required for forwarded ports, shared folders, host only
    servidor: networking, and more. If SSH fails on this machine, please install
    servidor: the guest additions and repackage the box to continue.
    servidor:
    servidor: This is not an error message; everything may continue to work properly,
    servidor: in which case you may ignore this message.
==> servidor: Setting hostname...
==> servidor: Configuring and enabling network interfaces...
==> servidor: Rsyncing folder: /home/atixlibre/vagrant/multiple/ => /vagrant
==> servidor: Machine already provisioned. Run `vagrant provision` or use the `--
provision`
==> servidor: flag to force provisioning. Provisioners marked to run always will still
run.
==> servidor: Machine 'servidor' has a post `vagrant up` message. This is a message
==> servidor: from the creator of the Vagrantfile, and not from Vagrant itself:
==> servidor:
==> servidor: Vanilla Debian box. See https://app.vagrantup.com/debian for help and bug
reports
```

Prueba de conexión

```
$ vagrant ssh cliente
```

```
~$ mysql -uatix -p -h192.168.33.11
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.26-MariaDB-0+deb9u1 Debian 9.1

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Creación de Box personal

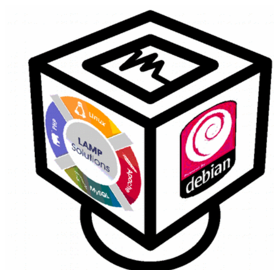
Muchas veces nos encontramos con la necesidad de publicar nuestro entorno de trabajo, o poner a disposición de nuestro equipo o de forma pública, las funcionalidades configuradas en nuestra máquina virtual, en estos casos es necesario tener la posibilidad de poder crear nuestra propia Box con todas las particularidades de nuestro proyecto y ponerla a disposición de terceros.

Procedimiento de creación de una Box

A continuación detallaremos el procedimiento para crear nuestra propia Box.

1. Crear una nueva máquina virtual
2. Instalar el VirtualBox Guest Additions
3. Configurar autenticación y la relación de confianza
4. Aprovisionar la máquina virtual
5. Exportar la máquina virtual
6. Probar la nueva Box

Entorno de demostración



Para realizar una demostración sencilla y comprensiva, procedemos a crear una máquina virtual, que en base a la distribución Debian strech, procederemos a instalar un entorno LAMP .

1. Crear una nueva máquina virtual

La creación de una nueva máquina virtual, se la realiza siguiendo el procedimiento habitual, cabe decir: disponiendo de la imagen ISO del sistema operativo, personalizando el mismo, e instalando los programas necesarios. Para nuestro ejemplo instalaremos una distribución Debian 9.4.

2. Instalar el VirtualBox Guest Additions

La instalación de virtual guest additions permitirá entre muchas cosas, compartir carpetas, y periféricos entre la máquina virtual y la máquina física.

Actualizar el sistema operativo

```
$ apt update
$ apt upgrade
```

Instalar los pre-requisitos

Debian

```
$ apt install linux-headers-$(uname -r) build-essential
$ apt-get install build-essential module-assistant
```

Centos 7

```
$ yum update kernel*
$ rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
$ yum install gcc kernel-devel kernel-headers dkms make bzip2 perl
```

o

```
$ yum groupinstall -y "Development Tools"
$ yum install -y dkms bzip2 perl
```

Montar el ISO de VirtualBox Guest Additions

Ejecutar el VirtualBox Guest Additions

```
$ VboxLinuxAdditions.run
```

3. Configurar autenticación y la relación de confianza

Creación del usuario

```
$ useradd -m vagrant
```

Cambiar la clave de root y vagrant

Por compatibilidad de las Box de vagrant, el usuario debe tener una contraseña "vagrant" tanto para root como para el usuario vagrant

```
$ passwd root
$ passwd vagrant
```

Relación de confianza

```
$ su - vagrant
$ mkdir .ssh
$ wget https://raw.githubusercontent.com/mitchellh/vagrant/master/keys/vagrant.pub
-O .ssh/authorized_keys
$ chmod 700 .ssh
$ chmod 600 .ssh/authorized_keys
```

Conceder permisos al usuario vagrant

Con esto existe la posibilidad de configurar la red, instalar software, montar carpetas compartidas.

```
$ visudo
vagrant ALL=(ALL) NOPASSWD: ALL
Defaults:vagrant !requiretty
```

Habilitar ssh para arranque automático dentro la máquina virtual

```
$ sudo systemctl enable ssh
$ sudo systemctl start ssh
```

4. Aprovisionar la máquina virtual

Para nuestro ejemplo, básicamente realizaremos la instalación de un entorno LAMP

```
$ apt install mariadb-server mariadb-client
$ apt install apache2
$ apt install php7.0 libapache2-mod-php7.0
```

5. Exportar la máquina virtual

Antes de exportar la máquina virtual, es importante hacer una limpieza de archivos temporales de la máquina virtual y rellenar de ceros el espacio restante para reducir la compresión de la misma.

Debian

```
$ rm –rf /tmp/*
$ apt-get clean
$ apt-get clean
$ dd if=/dev/zero of=/EMPTY bs=1M
$ rm -f /EMPTY
$ cat /dev/null > ~/.bash_history && history -c && exit
```

Centos 7

```
$ yum -y install yum-utils
$ package-cleanup -y --oldkernels --count=1
$ yum -y autoremove
$ yum -y remove yum-utils
$ yum clean all
$ rm -rf /tmp/*
$ rm -f /var/log/wtmp /var/log/btmp
$ dd if=/dev/zero of=/EMPTY bs=1M
$ rm -f /EMPTY
$ cat /dev/null > ~/.bash_history && history -c
```

Empaquetar la Box

Para nuestro ejemplo la máquina virtual creada y personalizada se llama “atixlibredebian94” y esta basada en debian 9.4 y tiene instalado y configurado un entorno LAMP, el nombre de la Box resultante es “atixlibre.box”

```
$ vagrant package --output atixlibre.box --base "atixlibredebian94"
==> atixlibredebian94: Exporting VM...
==> atixlibredebian94: Compressing package to: /home/atixlibre/atixlibre.box
```

6. Prueba de la Box personalizada

Añadir la box creada

```
$ vagrant box add atixlibre atixlibre.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'atixlibre' (v0) for provider:
    box: Unpacking necessary files from: file:///home/atixlibre/atixlibre.box
==> box: Successfully added box 'atixlibre' (v0) for 'virtualbox'!
```

Inicializar la nueva Box

```
vagrant init
```

Configuración del Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

# All Vagrant configuration is done below. The "2" in Vagrant.configure
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|
  config.vm.box = "atixlibre"
  config.vm.hostname = "lamp.atixlibre.org"
  config.vm.network "forwarded_port", guest: 80, host: 8082
  config.vm.network "private_network", ip: "192.168.10.102"
  config.vm.post_up_message = "La aplicación esta disponible en
                               http://192.168.10.102/info.php"
  config.vm.provider "virtualbox" do |hw|
    hw.memory = 512
    hw.cpus = 2
  end
end
```

Ejecutar la box creada

```
$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Importing base box 'atixlibre'...
==> default: Matching MAC address for NAT networking...
==> default: Setting the name of the VM: atixlibre_default_1530739760230_92871
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
    default: Adapter 2: hostonly
==> default: Forwarding ports...
    default: 80 (guest) => 8082 (host) (adapter 1)
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Running 'pre-boot' VM customizations...
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
    default: Warning: Connection reset. Retrying...
    default:
    default: Vagrant insecure key detected. Vagrant will automatically replace
    default: this with a newly generated keypair for better security.
    default:
    default: Inserting generated public key within guest...
    default: Removing insecure key from the guest if it's present...
    default: Key inserted! Disconnecting and reconnecting using new SSH key...
==> default: Machine booted and ready!
==> default: Checking for guest additions in VM...
==> default: Setting hostname...
==> default: Configuring and enabling network interfaces...
==> default: Mounting shared folders...
    default: /vagrant => /home/atixlibre/vagrant/atixlibre
```

Acceso al entorno LAMP

Desde la maquina física, mediante el uso de browser accedemos a **`http://192.168.10.102/info.php`** para verificar que el entorno LAMP este funcionando correctamente.

192.168.10.102/info.php   Buscar

PHP Version 7.0.27-0+deb9u1

php

System	Linux lamp 4.9.0-6-amd64 #1 SMP Debian 4.9.88-1+deb9u1 (2018-05-07) x86_64
Build Date	Jan 5 2018 13:51:52
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012

Conclusiones

- El uso de herramientas como Vagrant, no sólo nos permite automatizar la creación de una simple máquina virtual, sino más al contrario, nos permite crear verdaderos entornos de desarrollo o test, dentro de los cuales pueden existir varias Box, cada una personalizada de distinta forma, con su propia instalación de paquetes y configuración, pero con la posibilidad de poder interactuar entre todas y brindar un conjunto de servicios integrados.
- La creación de entornos simples o múltiples permite probar nuestro proyectos de desarrollo, como si se tratase de un verdadero entorno de producción.
- La posibilidad de poder crear nuestras propias Box, nos da la posibilidad de poder distribuir o transportar nuestros entornos de desarrollo o test totalmente personalizados, listos para su uso.

Agradecimiento

Un profundo y sincero agradecimiento al Ing. Esteban Saavedra, por haberme impulsado y colaborado en la elaboración de este artículo, siendo el coautor del mismo, por su actitud altruista hacia la formación/capacitación y por su incansable labor en impulsar el uso de tecnologías libres en todos los ámbitos.

Referencias

[1] http://www.vagrantup.com



Gabriela Antezana
Desarrollador
gabi.paola.antezana@gmail.com
BOLIVIA



Arduino

Aprendiendo Róbotica II

Actualmente, el mundo atraviesa por una gran ola de cambios tecnológicos donde el uso de las tecnologías libres se está difundiendo más y más y llegando a limites nunca antes vistos, razón por la cual las personas deben adaptarse y aprender día a día de la realidad que están viviendo.

Una tecnología que ha copado la atención de grandes y pequeños en todos los ámbitos es la utilización de hardware libre, que permite crear entornos automatizados, prototipos, robótica educativa entre otros.

Elementos importantes de desarrollo

Existen varios elementos importantes que se deben considerar al momento de desarrollar proyectos basados en Arduino, tomando en cuenta elementos conceptuales de hardware y software.

Gestor de Arranque (Bootloader)

1. El bootloader es un pequeño programa que se encuentra almacenado en la memoria flash de la placa Arduino.
2. Es el primer programa que se ejecuta en el procesador cuando enciendes o haces un reset del Arduino.
3. Es el que se encarga de cargar el Sketch (programa del usuario) cargado previamente.

EEPROM

Arduino dispone de una memoria (EEPROM), es decir, una memoria no volátil (como la de un disco duro) en la que puedes almacenar datos sin preocuparte de perderlos cuando le quites la alimentación a tu Arduino.

ICSP

ICSP (In-Circuit Serial Programming), es la habilidad de algunos dispositivos lógicos programables, microcontroladores y otros circuitos electrónicos, de ser programados mientras están instalados en un sistema completo, en lugar de requerir que el chip sea programado antes de ser instalado dentro del sistema.

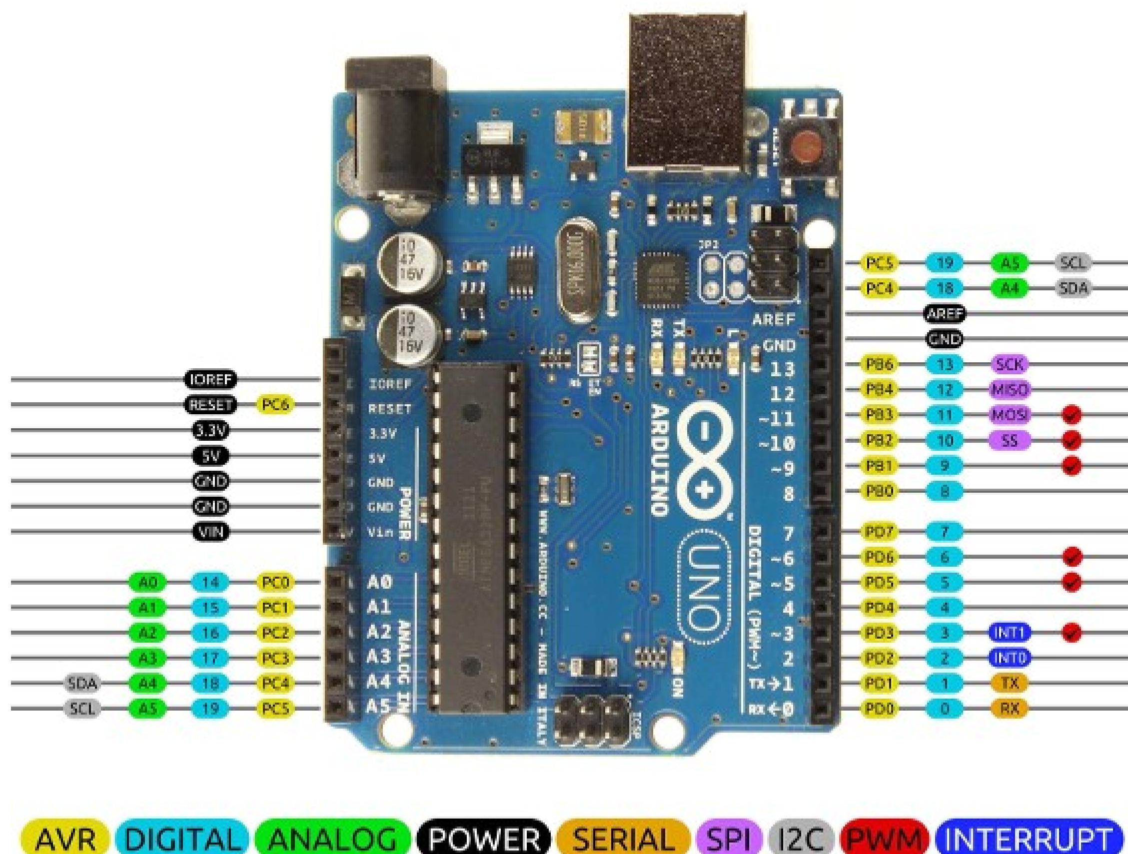
Interrupciones - Timers

Arduino dispone de dos maneras de alertar que ha ocurrido cierto evento:

- **Interrupciones de hardware**: que permite alertar que ha ocurrido cierto evento o que la ejecución de cierto programa esta afectando alguno de los puertos.
- **Timers**: que permite alertarnos cuando haya transcurrido un tiempo preciso previamente programado.

Distribución de pines

Si bien no existe restricciones y podemos utilizar cualquier Pin para casi cualquier función, la placa de Arduino UNO esta diseñada para facilitarnos las cosas y ya viene con recomendaciones de que pines usar para cada acción.



Pines de Entrada y Salida

Arduino UNO posee 14 pines digitales, de los cuales todos pueden ser utilizados tanto como entrada o como salida, eso dependerá de la programación que realicemos y como tengamos definidos los pines.

Pines de comunicación SPI

Es común que los microcontraladores cuenten con arquitectura para soportar la comunicación mediante SPI. Arduino UNO cuenta con cuatro pines destinados a esta tarea

- **Pin 10 (SS)**: (Chip Select o Slave Select), habilita el integrado hacia el que se envían los datos, puede ser opcional.
- **Pin 11 (MOSI)**: (Master Output Slave Input): Transmisión de datos hacia el otro integrado.
- **Pin 12 (MISO)**: (Master Input Slave Output): Es la señal de entrada a nuestro dispositivo, por aquí se reciben los datos desde el otro integrado.
- **Pin 13 (SCK)**: Señal de reloj del bus. Esta señal rige la velocidad a la que se transmite cada bit.

Pines PWM

En Arduino UNO, los pines 3,5,6,9,10,11 están preparados para brindar una salida de PWM de ocho bits, mediante la función analogWrite().

Interrupciones Externas

Los pines 2 y 3 son generalmente utilizados para generar interrupciones externas, de las que se conocen como interrupciones de hardware.

Pin 13

Este pin es el único que tiene un led conectado en serie a la entrada, así que si queremos testear algo mediante el brillo de un led, podemos utilizar este pin 13

Pin AREF

Este pin mantiene una tensión de referencia que es útil para realizar conversiones analógicas a digitales. Para esto existen una serie de librerías para realizar esta tarea de forma simple.

Programación

Programar con las placas Arduino es bastante simple, esta placa está pensada para simplificar las tareas más complejas mediante librerías disponibles para su uso, este tipo de placas utiliza un lenguaje de programación que está basado en C++, un lenguaje simple y eficaz.

Una vez definidos algunos conceptos operativos y funcionales de la placa Arduino, es hora de ingresar a los elementos que le dotarán de mayor dinamismo a nuestros experimentos, como es la programación.

Estructura de un programa

La estructura básica de un programa en Arduino es bastante simple y se organiza en al menos dos partes o funciones que encierran bloques de declaraciones.

```
void setup()
{
  instrucciones;
}
void loop()
{
  instrucciones;
}
```

Setup()

La función setup debe contener la declaración o inicialización de cualquier variable al comienzo del programa.

Es la primera función a ejecutar en el programa, es ejecutada una vez y es usada para asignar valores o inicializar las comunicaciones.

```
void setup()
{
  pinMode(pin, OUTPUT); //ajusta 'pin'
como salida
}
```

loop()

La función loop incluye el código que se ejecuta repetitivamente leyendo entradas, activando salidas, realizando cálculos y/o procesos, etc. Esta función es el núcleo de todos los programas Arduino y hace la mayor parte del trabajo.

```
void loop()
{
  digitalWrite(pin, HIGH); //Activa 'pin'
  delay(1000);//espera un segundo
  digitalWrite(pin, LOW); //Desactiva
                           'pin'
  delay(1000);//espera un segundo
}
```

Elementos de un programa

Para realizar una mejor programación, es necesario saber definir los elementos que intervienen en un programa.

Variables

Son elementos que pueden almacenar valores que cambian continuamente en el transcurso del programa.



Robótica

```
int contador = 0;
//declara una variable llamada “contador”
y asigna el valor a 0

lectura = analogRead(2);
//ajusta el valor de la variable
“lectura” al valor del pin analógico 2
```

Constantes

Son elementos que no cambian de valor en el transcurso del programa.

```
#define LEDPIN 3
//hace uso de la clausula define para
definir el valor de una constante

const float PI = 3.14;
//hace uso de la clausula const para
definir el valor de una constante
```

Tipos de datos

Como en cualquier entorno o plataforma de desarrollo, Arduino permite hacer uso de una serie de tipos de datos.

byte

Byte almacena un valor numérico de 8 bits sin puntos decimales. Tienen un rango de 0 a 255.

```
byte a = 180;
```

int

Almacenan un valor de 16 bits con un rango de -32,768 a 32,767.

```
int b = 1500;
```

long

Almacena un valor de 32 bits con un rango de -2,146,483,648 a 2,147,483,647.

```
long c = 90000;
```

float

Almacenan valores de 32 bits con un rango de -3.4028235E+38 a 3.4028235E+38.

```
float pi = 3.14;
```

arrays

Un array es una colección de valores que son accedidos con un índice numérico.

```
int pares[] = {2,4,6,8,10};
```

Estructuras de control

Las estructuras de control permiten controlar el flujo de un programa, pudiendo establecer cierto flujo en base al cumplimiento o no de cierta condición selectiva o de repetición.

Estructura If

Permiten comprobar o verificar si cierta condición ha sido alcanzada.

```
if(condición)
{
  Instrucciones si la condición es
  verdadera;
}
else
{
  Instrucciones si la condición es falsa;
}
```

Estructuras Repetitivas

Estas estructuras permiten realizar una serie de repeticiones de un conjunto de instrucciones, de forma finita o en base a cierta condición lógica.

Existen la siguientes estructuras repetitivas:

- For
- While
- Do …. While

Estructura repetitiva For

Sintaxis

```
for(inicializacion; condicion; expresion)
{
  instrucciones;
}
```

Ejemplo

Encender y apagar un led 20 veces; para esto hacemos uso de una repetición de la variable i desde 0 hasta el 20 con incrementos de 1 en 1




```
for(int i=0; i<=20; i++)
{
  digitalWrite(13, HIGH);
  delay(250);
  digitalWrite(13, LOW);
  delay(250);
}
```

Estructura repetitiva While

Sintaxis

```
while(condición)
{
  instrucciones;
}
```

Ejemplo

El mismo ejercicio anterior pero haciendo uso de la estructura while

```
i=1;
while(i<=20)
{
  digitalWrite(13, HIGH);
  delay(250);
  digitalWrite(13, LOW);
  delay(250);
  i++;
}
```

Estructura repetitiva Do While

Sintaxis

```
do
{
  instrucciones
}
While (condición);
```

Ejemplo

```
i=0;
do
{
  digitalWrite(13, HIGH);
  delay(250);
  digitalWrite(13, LOW);
  delay(250);
  i++;
}
while(x <= 20);
```

Funciones

Una función es un bloque de código que tiene un nombre y un grupo de declaraciones e instrucciones con un objetivo específico, que se ejecutan cuando se la invocan.

Sintaxis

```
type NombreFuncion(parametros)
{
  instrucciones;
}
```

Ejemplo

```
int suma(int a, int b)
{
  int s;
  s=a+b;
  return s;
}
```

Función Delay()

Es una función destinada al manejo del tiempo o para generar algún retraso. No siempre es recomendable hacer uso de ella, ya que presenta algunas desventajas:

- No puedes hacer otra cosa mientras se está ejecutando.
- Es incompatible con las interrupciones.
- Aumenta el consumo del proyecto.
- No se pueden detectar eventos externos (interrupciones) e internos (interrupciones por timer)

Función Millis()

Esta función tiene un valor de retorno, nos devuelve el tiempo en mili segundos que transcurrido desde el inicio del programa hasta que se ha realizado la función, generalmente este es un valor grande dependiendo del tiempo entre la ejecución y la siguiente.

La ventaja de esta función es que nos permite obtener los mismos resultados que con la función delay pero podemos hacer cualquier otra cosa mientras este evento no se haya compilado.

Las ventajas del uso de esta función son:

- Es compatible con las interrupciones.
- Podemos hacer otra cosa mientras esperamos que un evento ocurra.
- No generamos carga innecesaria a nuestro procesador.

Instalación de librerías de terceros

Una forma de extender las capacidades de software que pose Arduino, es la inclusión de librerías realizadas por terceros. Teniendo en cuenta que una librería es una colección de código con cierto propósito que puede ser reutilizado en varios programas.

Algunos ejemplos:

```
#include <Dhcp.h>
#include <EthernetServer.h>
#include <EthernetClient.h>
#include <EthernetUdp.h>
#include <Ethernet.h>
#include <Dns.h>
```

Dentro el lenguaje de programación que facilita Arduino, existe la posibilidad de cada usuario pueda desarrollar sus propias librerías, recordemos que el uso de librerías es una forma de estructurar y encapsular adecuadamente los programas, haciéndalos más entendibles y reutilizar código ya desarrollado.

Señales Digitales

Una señal digital es una variación de voltaje entre -Vcc a +Vcc sin pasar por los valores intermedios. Por lo tanto, una señal digital dispone solo de dos estados.

- Al valor inferior -Vcc le asociamos LOW o '0'
- Al valor superior +Vcc le asociamos HIGH o '1'.

Sin embargo en el mundo físico las referencias de tensión realmente son continuas. El proceso de lectura digital es un proceso de discretización de una señal analógica.



Conexión de E/S digitales

Para poder asignarle o recuperar el valor de un determinado PIN de la placa de Arduino, es preciso determinar el tipo de conexión (INPUT/OUTPUT).

```
int pin_entrada = 2;
int pin_salida = 0;
void setup() {
  pinMode(pin_entrada, INPUT);
  //definir pin como entrada

  pinMode(pin_salida, OUTPUT);
  //definir pin como salida
}
```

Lectura / Escritura de señales digitales

Función digitalRead(pin)

Lee el valor desde un pin digital específico. Devuelve un valor HIGH o LOW. El pin puede ser especificado con una variable o una constante.

Ejemplo:

```
int pin 5
valor = digitalRead(pin);
```

Función digitalWrite(pin)

Introduce un nivel alto (HIGH) o bajo (LOW) en el pin digital especificado. El pin puede ser especificado con una variable o una constante.

Ejemplo:

```
int pin 6;
digitalWrite(pin, HIGH);
Delay(1000);
digitalWrite(pin, LOW);
```

Señales Analógicas

- Una señal analógica es una magnitud que puede tomar cualquier valor dentro de un intervalo –Vcc y + Vcc.
- Pueden leer valores de tensión de 0 a 5 Voltios con una resolución de 1024 (10 bits), podría decirse es capaz de detectar variaciones en el nivel de la señal de entrada de casi 5 mV.



Lectura / Escritura de señales analógicas

Función analogWrite(pin, valor)

Escribe un valor pseudo-analógico usando modulación por ancho de pulso (PWM) en un pin de salida marcado como PWM. Esta función está activa para los pines 3, 5, 6, 9, 10, 11.

Ejemplo:

```
int pin 6;
analogWrite(pin, 214);
//214 equivale a 4.2V
```

Función analogRead(pin)

Lee el valor desde el pin analógico especificado con una resolución de 10 bits. Esta función solo funciona en los pines analógicos (0-5). El valor resultante es un entero de 0 a 1023.

Los pines analógicos, a diferencia de los digitales no necesitan declararse previamente como INPUT o OUTPUT.

Ejemplo:

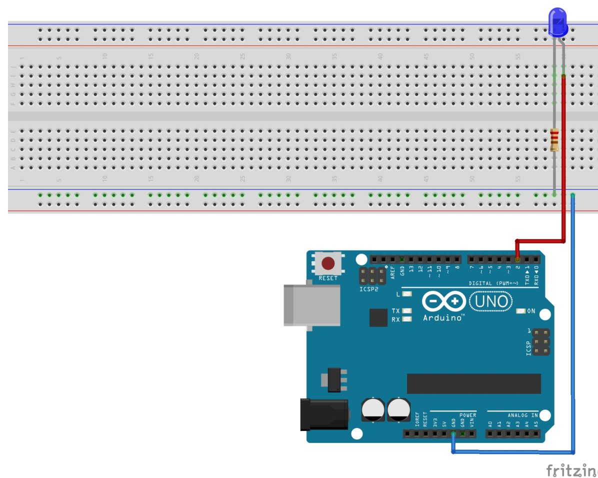
```
int pin 5
valor = analogRead(pin);
```

Anímate a dar el primer paso

Todo comienza con pequeñas pruebas, desde los más grandes inventos hasta los más pequeños. Anímate a hacer secuencias de programación desde tu hogar y tu comodidad; recuerda no necesitas ser un experto para iniciar con Arduino. A continuación te dejo algunos de mis primeros programas, utilízalos de guía, pruébalos y haz que tu imaginación vuele hasta las estrellas.

Encender un Led

Uno de los programas con lo que debes iniciar Arduino es encender y apagar un led, haciendo este pequeño proyecto iras creciendo poco a poco. Y recuerda tu imaginación llega hasta donde tu quieras.



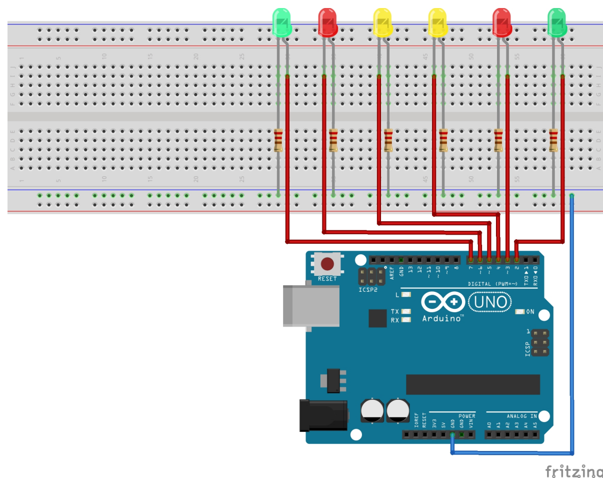

```
int LED=2;
int t=1000;
void setup()
{
  pinMode (LED, OUTPUT);

}
void loop()
{
  digitalWrite (LED, HIGH);
  delay (t);
  digitalWrite (LED, LOW);
}
```



Secuencia de Leds

Encender y apagar un led fue el primer paso con el que todos iniciamos a programar, pero para poder crecer, debemos a animarnos a más poco a poco. Si tú tienes las ganas de programar, muy pronto tú serás el que esté compartiendo conocimiento a nuevas personas.



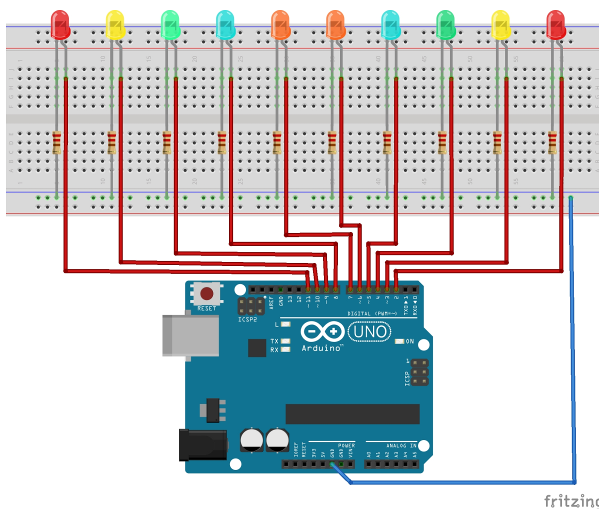

```
int VERDE=2;
int ROJO=3;
int AMARILLO=4;
int AMARILLO2=5;
int ROJO2=6;
int VERDE2=7;
int t=1000;

void setup()
{
  pinMode (VERDE, OUTPUT);
  pinMode (ROJO, OUTPUT);
  pinMode (AMARILLO, OUTPUT);
  pinMode (AMARILLO2, OUTPUT);
  pinMode (ROJO2, OUTPUT);
  pinMode (VERDE2, OUTPUT);
}

void loop()
{
  digitalWrite (VERDE, HIGH);
  digitalWrite (VERDE2, HIGH);
  delay (t);
  digitalWrite (VERDE, LOW);
  digitalWrite (VERDE2, LOW);
  delay (t);
  digitalWrite (ROJO, HIGH);
  digitalWrite (ROJO2, HIGH);
  delay (t);
  digitalWrite (ROJO, LOW);
  digitalWrite (ROJO2, LOW);
  delay (t);
  digitalWrite (AMARILLO, HIGH);
  digitalWrite (AMARILLO2, HIGH);
  delay (t);
  digitalWrite (AMARILLO, LOW);
  digitalWrite (AMARILLO2, LOW);
}
```

Segunda secuencia de Leds

En este ejemplo mostraremos, la gran ventaja que tiene el utilizar las estructuras repetitivas, aspecto que nos demuestra como podemos hacer nuestros programas de una forma más óptima. Y con un número de lineas considerablemente menor.



Sin el uso de estructuras repetitivas.

```
int a=2;
int b=3;
int c=4;
int d=5;
int e=6;
int f=7;
int g=8;
int h=9;
int i=10;
int j=11;
int x=1000;
void setup()
{
  pinMode (a, OUTPUT);
  pinMode (b, OUTPUT);
  pinMode (c, OUTPUT);
  pinMode (d, OUTPUT);
  pinMode (e, OUTPUT);
  pinMode (f, OUTPUT);
  pinMode (g, OUTPUT);
  pinMode (h, OUTPUT);
  pinMode (i, OUTPUT);
  pinMode (j, OUTPUT);
}
```




```
void loop()
{
  // Primera vuelta
  digitalWrite (a, HIGH);
  digitalWrite (j, HIGH);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  delay (x);
  digitalWrite (b, HIGH);
  digitalWrite (i, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (c, HIGH);
  digitalWrite (h, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (d, HIGH);
  digitalWrite (g, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (e, HIGH);
  digitalWrite (f, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (500);
```

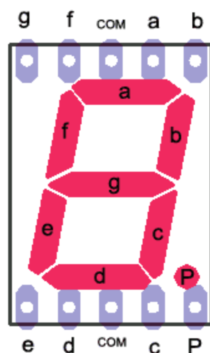
```
  // Segunda vuelta
  digitalWrite (e, HIGH);
  digitalWrite (f, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (d, HIGH);
  digitalWrite (g, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (c, HIGH);
  digitalWrite (h, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (b, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (i, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (b, HIGH);
  digitalWrite (i, HIGH);
  digitalWrite (a, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (j, LOW);
  delay (x);
  digitalWrite (a, HIGH);
  digitalWrite (j, HIGH);
  digitalWrite (b, LOW);
  digitalWrite (c, LOW);
  digitalWrite (d, LOW);
  digitalWrite (e, LOW);
  digitalWrite (f, LOW);
  digitalWrite (g, LOW);
  digitalWrite (h, LOW);
  digitalWrite (i, LOW);
  delay (x);
}
```

Con el uso de estructuras repetitivas

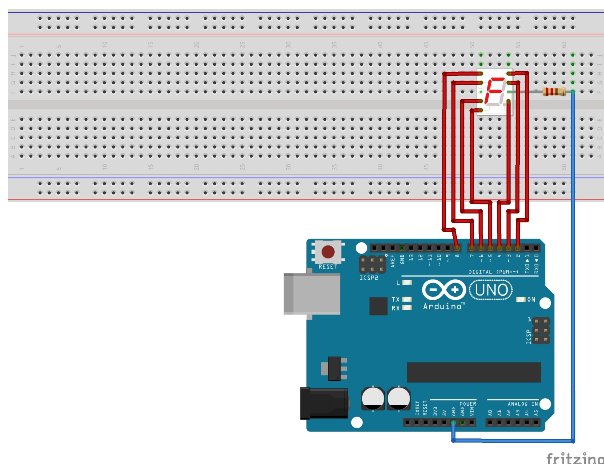
```
int i;
void setup()
{
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}
void loop()
{
 //Primera vuelta
  for(i=11; i>6; i--)
  {
    digitalWrite(i,HIGH);
    digitalWrite(13-i,HIGH);
    delay(10);
    digitalWrite(i,LOW);
    digitalWrite(13-i,LOW);
  }
 //Segunda Vuelta
  for(i=7; i<=11; i++)
  {
    digitalWrite(i,HIGH);
    digitalWrite(13-i,HIGH);
    delay(10);
    digitalWrite(i,LOW);
    digitalWrite(13-i,LOW);
  }
}
```

Display de 7 secuencias

La imaginación nos puede llevar hasta las estrellas, si nos animamos lo podemos hacer todo. Alguna vez te preguntaste como funciona un semáforo o un reloj digital, bueno si te animas puedes hacerlo desde la comodidad de tu hogar con un Arduino, con un display de 7 segmentos y una resistencia.



Encender secuencialmente todos los segmentos del display.



Sin el uso de estructuras repetitivas

```
 int a=2;
 int b=3;
 int c=4;
 int d=5;
 int e=6;
 int f=7;
 int g=8;
 int p=9;

void setup()
{
 pinMode(a,OUTPUT);
 pinMode(b,OUTPUT);
 pinMode(c,OUTPUT);
 pinMode(d,OUTPUT);
 pinMode(e,OUTPUT);
 pinMode(f,OUTPUT);
 pinMode(g,OUTPUT);
 pinMode(p,OUTPUT);
}

void loop()
{
  digitalWrite(a,HIGH);
  delay(100);
  digitalWrite(a,LOW);
  delay(100);
  digitalWrite(b,HIGH);
  delay(100);
  digitalWrite(b,LOW);
  delay(100);
  digitalWrite(c,HIGH);
  delay(100);
  digitalWrite(c,LOW);
  delay(100);
  digitalWrite(d,HIGH);
  delay(100);
  digitalWrite(d,LOW);
  delay(100);
  digitalWrite(e,HIGH);
  delay(100);
```




```
  digitalWrite(e,LOW);
  delay(100);
  digitalWrite(f,HIGH);
  delay(100);
  digitalWrite(f,LOW);
  delay(100);
  digitalWrite(g,HIGH);
  delay(100);
  digitalWrite(g,LOW);
  delay(100);
  digitalWrite(p,HIGH);
  delay(100);
  digitalWrite(p,LOW);
  delay(100);
}
```

Con el uso de estructuras repetitivas

```
int i;
void setup()
{
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
}
void loop()
{
  for(i=2; i<=9; i++)
  {
    digitalWrite(i,HIGH);
    delay(100);
    digitalWrite(i,LOW);
  }
}
```

Contador con display de 7 segmentos

Mostrar los dígitos del 1 al 9 de forma repetitiva con cierto intervalo de tiempo.

```
int pausa=1000;
void setup()
{
  // Asignación de las salidas digitales
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
}
```

```
//Función que enciende o apaga los
  segmentos según corresponda

void display (int a, int b, int c, int d,
int e, int f, int g)
{
  //Se reciben 7 variables asignadas a
    cada segmento
  digitalWrite (2,a);
  digitalWrite (3,b);
  digitalWrite (4,c);
  digitalWrite (5,d);
  digitalWrite (6,e);
  digitalWrite (7,f);
  digitalWrite (8,g);
}

void loop()
{
  display (1,1,1,1,1,1,0); //escribe 0
  delay(pausa);
  display (0,1,1,0,0,0,0); //escribe 1
  delay(pausa);
  display (1,1,0,1,1,0,1); //escribe 2
  delay(pausa);
  display (1,1,1,1,0,0,1); //escribe 3
  delay(pausa);
  display (0,1,1,0,0,1,1); //escribe 4
  delay(pausa);
  display (1,0,1,1,0,1,1); //escribe 5
  delay(pausa);
  display (1,0,1,1,1,1,1); //escribe 6
  delay(pausa);
  display (1,1,1,0,0,0,0); //escribe 7
  delay(pausa);
  display (1,1,1,1,1,1,1); //escribe 8
  delay(pausa);
  display (1,1,1,0,0,1,1); //escribe 9
  delay(pausa);
}
```

Referencias

[1] http://www.Arduino.cc



Stephanie Saavedra
Entusiasta de Robótica
stephanie.saavedra.ayarde@gmail.com
BOLIVIA

4 Creando Epub con OpenOffice

Los archivos EPUB (Electronic publication - Publicación electrónica) están definidos por un formato redimensionable de código abierto para leer textos e imágenes, muy utilizado hoy en día para poder leer libros EBOOK en cualquier tipo de dispositivo (smartphones, tablets, laptops, etc).

Introducción

La portabilidad de los archivos EPUB, al poder redimensionar el tamaño de las letras e imágenes hacen que este formato de archivos sea muy utilizado para la distribución de libros, ya que permite que el contenido se pueda adaptar a distintos tipos de pantallas y tamaños de letras sin perder la calidad.

En este artículo utilizaremos el software OpenOffice-Writer con la extensión WRITER2EPUB para poder crear documentos en formato EPUB y conoceremos también la aplicación CALIBRE para la lectura de ebooks.



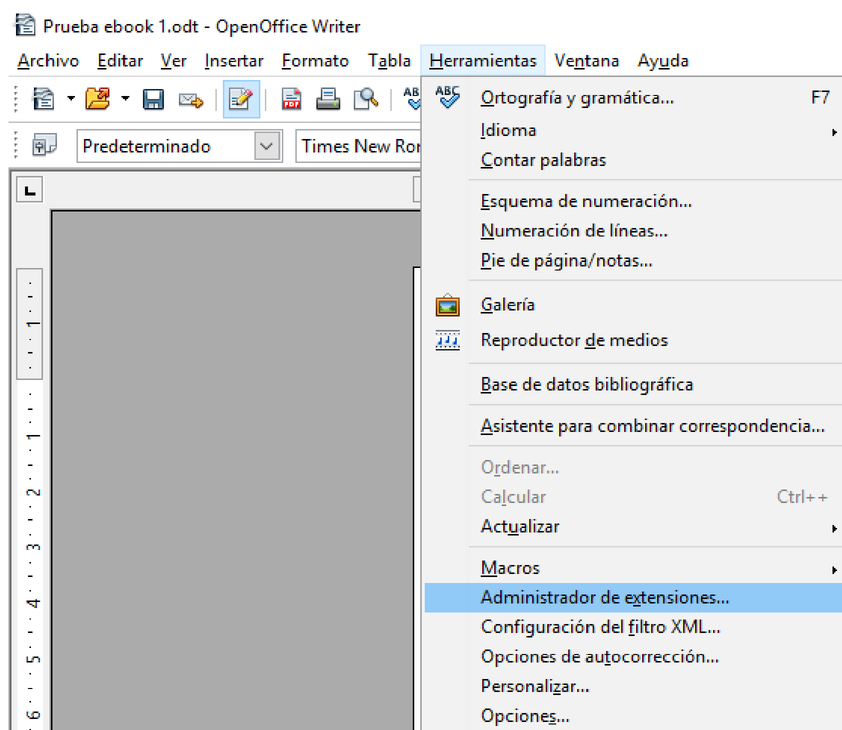
Entre las características de un archivo EPUB están:

- Código abierto.
- Formato estándar compatible con los distintos software o hardware de eReaders.
- Codificación Unicode lo que garantiza que no habrá problemas de multilenguaje.
- Formato dinámico y expandible que admite imágenes.
- Soporta imágenes y vídeo.

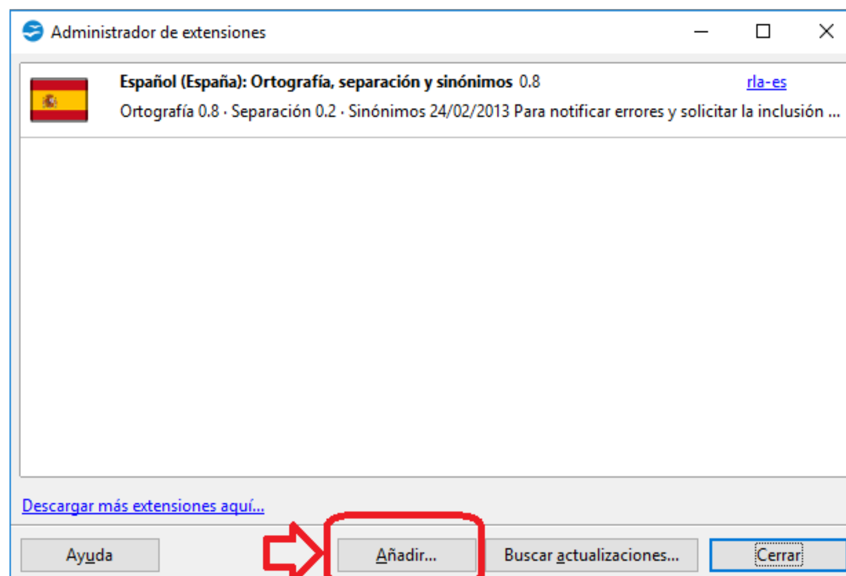
Instalación y ejecución

Como requisito debemos tener instalado una versión de OpenOffice Writer 3 o superior. A continuación descargamos la extensión Writer2Epub de la siguiente URL: **`http://writer2epub.it/en/download/`** .

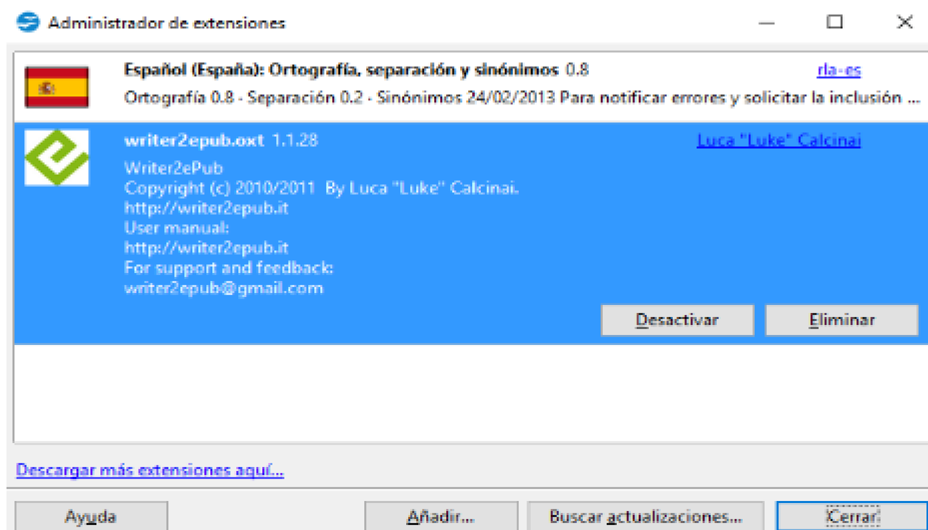
Para la instalación simplemente debemos cargar la extensión en OpenOffice-Writer, para lo cual abrimos en el menú HERRAMIENTAS la opción ADMINISTRADOR DE EXTENSIONES.



Elegimos Añadir:



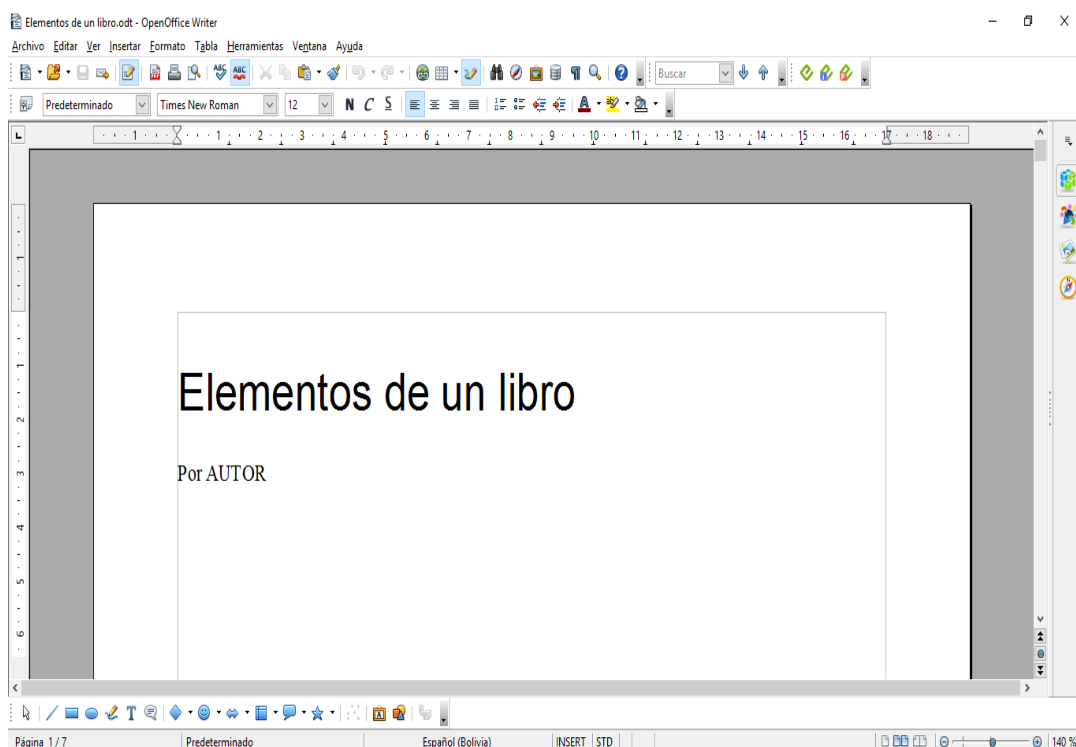
Luego añadimos la extensión writer2epub.oxt



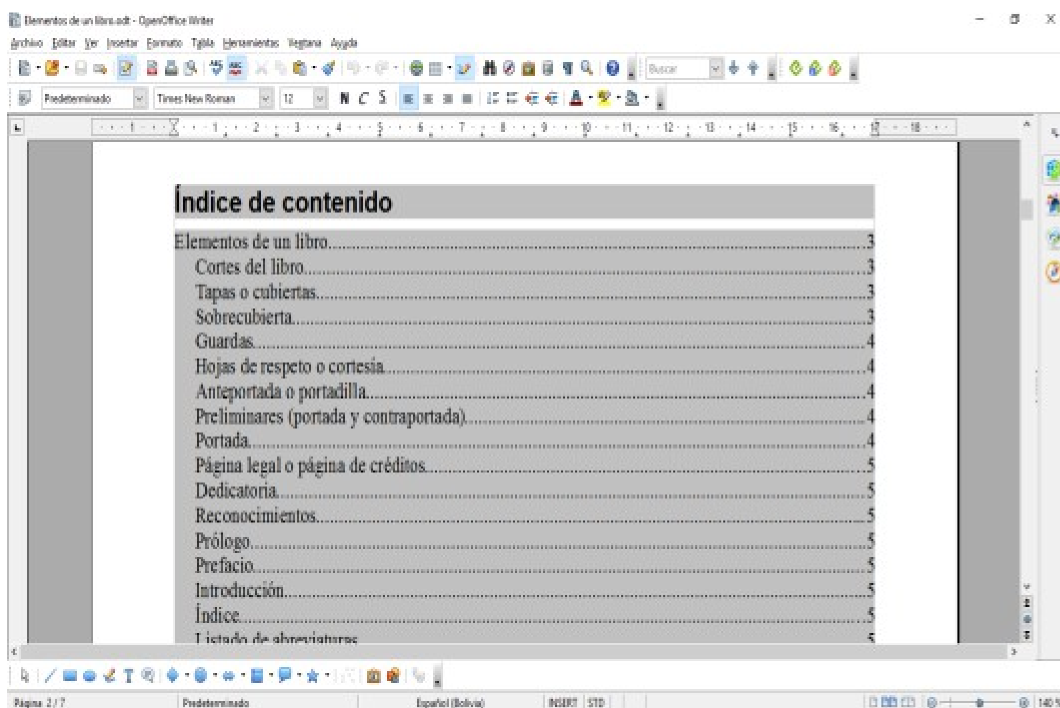
Ejemplo

A continuación, explicamos en un ejemplo como podemos crear un archivo EPUB:

1. Tomaremos un documento ya creado en OpenOffice-Writer, archivo .odt. Sugerimos que el documento contenga un ÍNDICE DE CONTENIDO para poder utilizar de una manera óptima el EPUB: Ejemplo de un documento con índice llamado “Elementos de un Libro”:



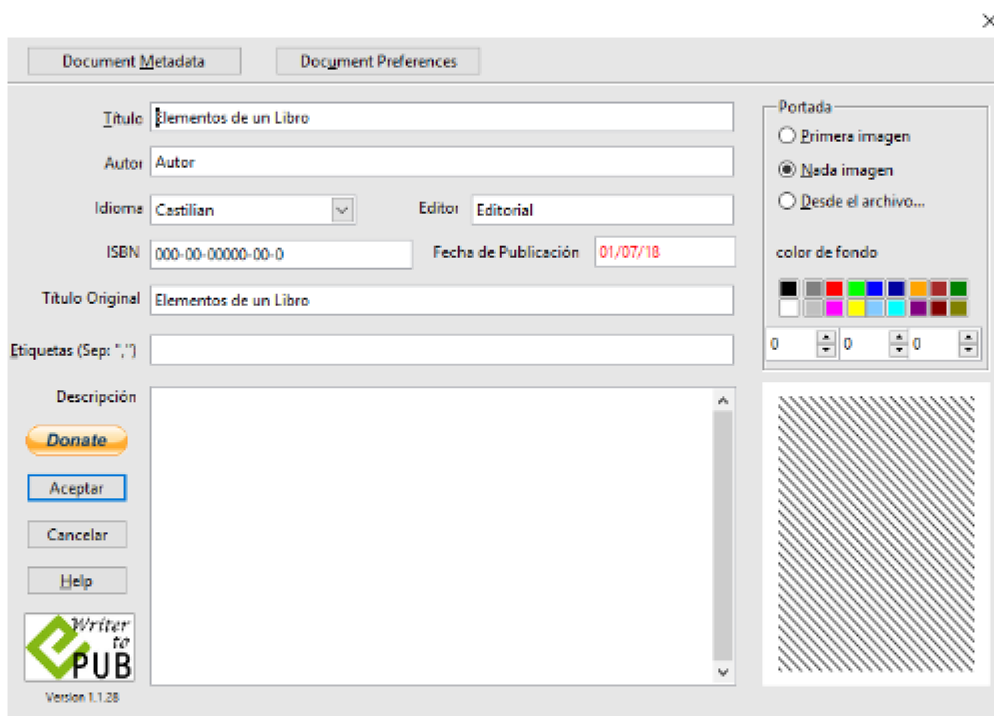




2. Elegimos en la barra de tareas el icono de la extensión de writer2epub:



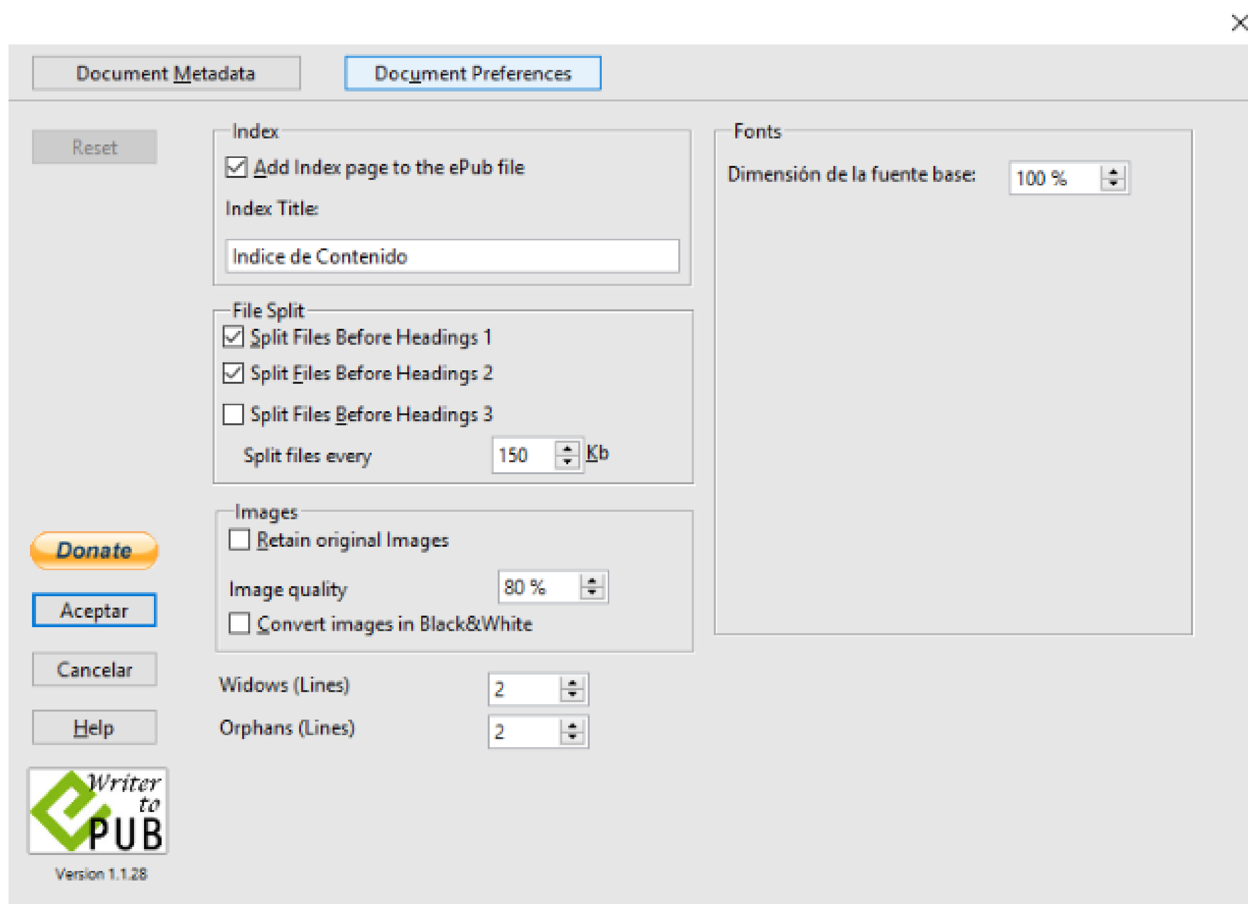
3. Se abrirá la siguiente ventana de configuración del EPUB, en la cual llenamos los datos del documento o libro que estamos convirtiendo en archivo EPUB:



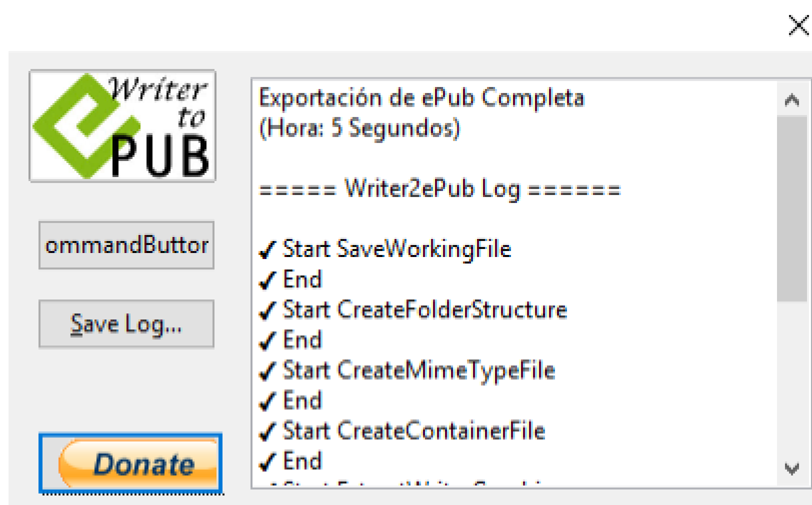
Donde ISBN es un código normalizado internacional para libros (International Standard Book Number).

En la PORTADA podemos insertar una imagen o dejarlo en blanco.

4. En la pestaña de DOCUMENT PREFERENCES, puedes incluir el ÍNDICE DE CONTENIDO:



5. Por último elegimos ACEPTAR para generar el archivo EPUB.

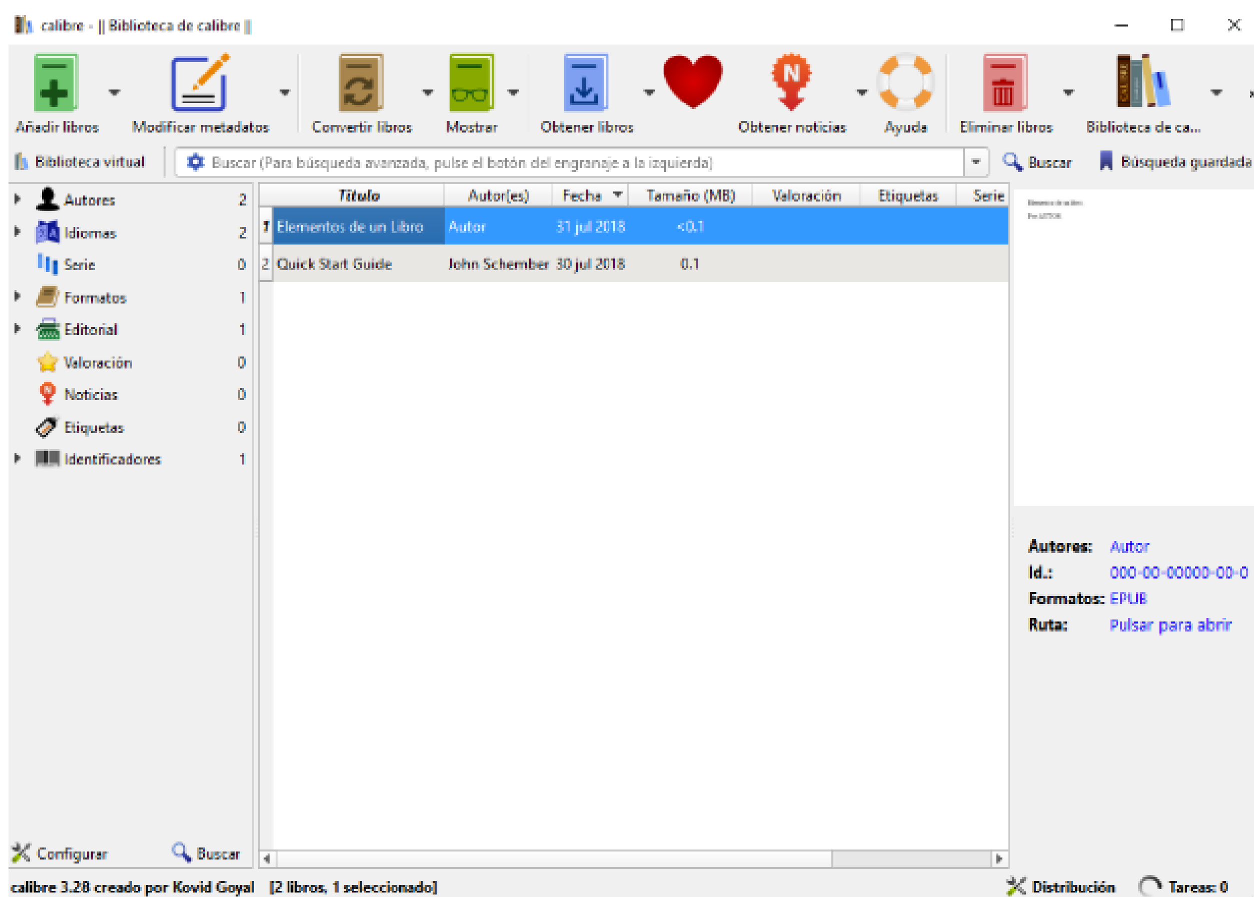


Lectura del documento Epub

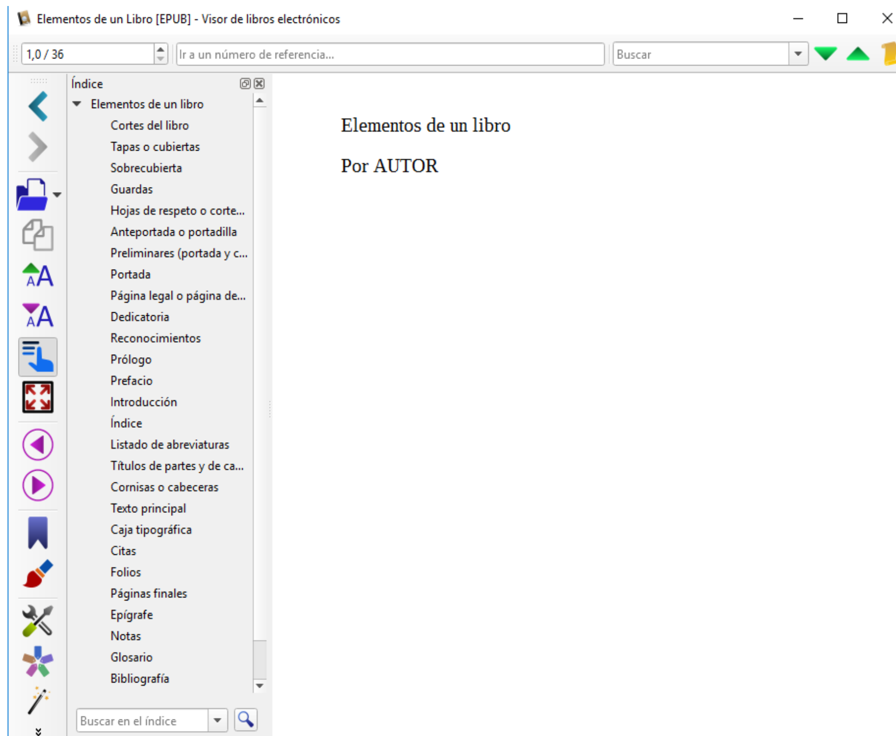
Para la lectura de documentos en formato EPUB existen varias alternativas, desde los dispositivos conocidos como eReaders, como por ejemplo el famoso KINDLE de Amazon, hasta diferentes aplicaciones para PC, Tablets o Smartphones, la aplicación que les recomiendo es CALIBRE que la pueden descargar de la siguiente dirección **`https://calibre-ebook.com/download_linux`**

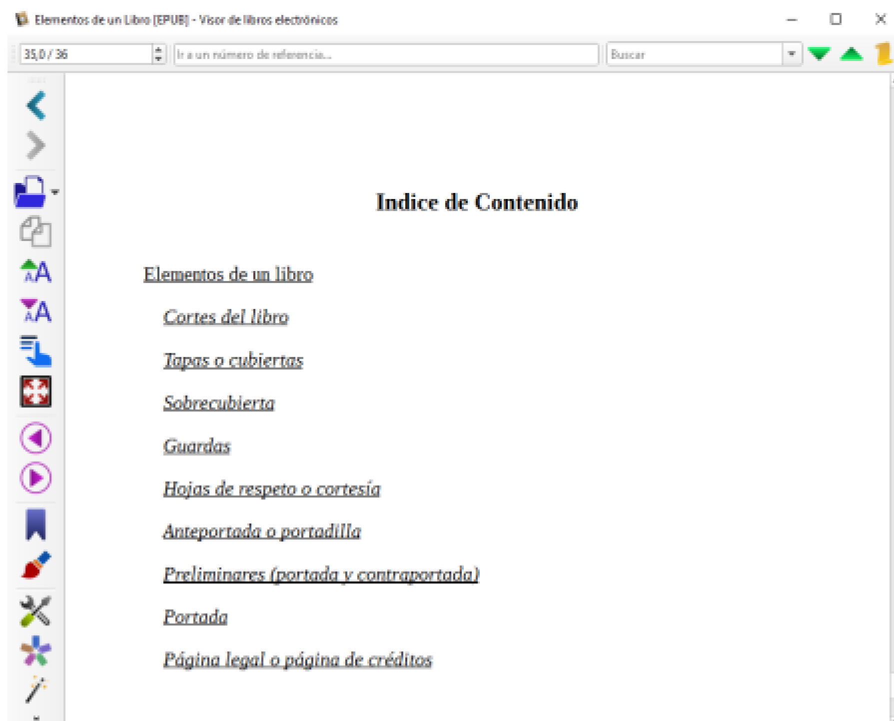
CALIBRE no sólo nos sirve para realizar la lectura de un documento EPUB, también nos permite editarlos y convertirlos en otros formatos de documentos digitales como ser AZW3, MOBI, PDF y muchos más.

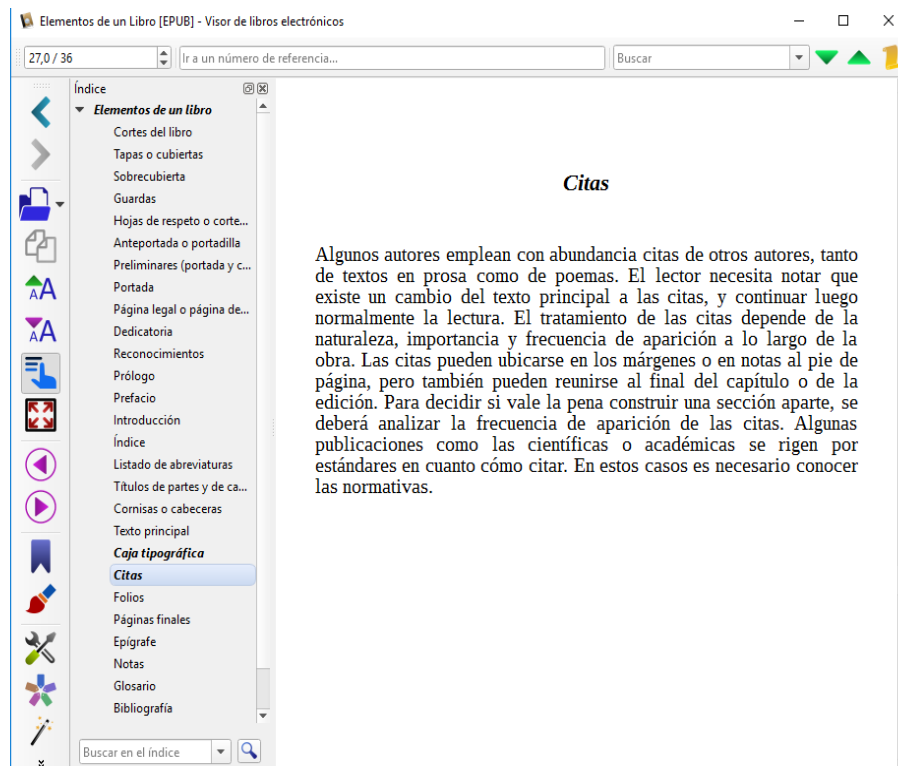
A continuación, las capturas de pantalla de CALIBRE:



Abrimos el archivo “Elementos de un libro” y podremos leer el documento, realizar marcaciones, notas, etc.







Conclusiones

- Este tipo de herramienta nos facilita el crear ebooks y los archivos generados son estándares y pueden ser utilizados en distintos dispositivos para su lectura.
- El uso de los libros digitales o ebooks tienen varias ventajas en comparación con los libros físicos. Los Ebooks no ocupan espacio físico, no se deterioran, son más baratos y rápidos de producir, pueden ser entregados casi al instante, no tienen ediciones agotadas y sobre todo se acomodan al lector.

Referencias

[1] http://writer2epub.it/

[2] https://calibre-ebook.com/



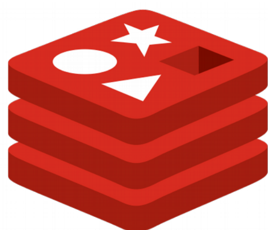
Renzo Martinez Pardo
Ingeniero de Sistemas Electrónicos
renzocmp@gmail.com
BOLIVIA



Redis
Bases de Datos en Memoria

En un mundo donde la velocidad de respuesta y procesamiento es crucial, la mayoría de las aplicaciones han optado por hacer uso de bases de datos en memoria, aspecto que se ha convertido en una tarea crucial, sobre todo en contextos donde se precise que el acceso a datos sea rápido, y muchas veces efímero.

Que es Redis



redis

Redis es un motor de base de datos en memoria, basado en el almacenamiento en tablas de hashes (clave/valor) pero que opcionalmente puede ser usada como una base de datos durable o persistente.

Con Redis, la mayoría de los datos se encuentran en memoria, haciendo que la búsqueda de información requerida a través de queries sea mucho más rápida y eficiente.

Gracias a su velocidad y facilidad de uso, Redis es una opción popular para aplicaciones web, móviles, juegos, de tecnología publicitaria y de IoT que requieren el mejor desempeño

Ventajas

- **Desempeño increíblemente rápido**: Al encontrarse los datos en memoria, el desempeño de las aplicaciones que hacen uso de los mismos es increíblemente rápido.
- **Estructuras de datos en memoria:** Permite almacenar y gestionar diversos tipos de datos. El tipo de datos fundamental es una cadena, que puede componerse de texto o datos binarios y tener un tamaño de hasta 512 MB.
- **Versatilidad y facilidad de uso**: Incorpora varias herramientas que facilitan y aceleran el desarrollo y las operaciones, incluídas Pub/Sub, para publicar mensajes en canales, que se entregan a suscriptores, lo que es ideal para sistemas de chat y mensajería; las claves TTL, que indican un tiempo de vida determinado, tras el que se eliminan a sí mismas
- **Replicación y persistencia**: Utiliza una arquitectura maestro-esclavo y admite la replicación asíncrona mediante la que los datos se replican en numerosos servidores esclavos.
- **Durabilidad**: Admite las snapshots de un momento determinado (copiando el conjunto de datos en disco) y la creación de un archivo de solo anexos (AOF) para almacenar cada uno de los cambios a los datos en disco a medida que estos se producen.
- **Compatibilidad de programación:** Los desarrolladores de Redis tienen a su disposición más de cien clientes de código abierto. Entre los lenguajes admitidos se encuentran Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, R, Go y muchos otros.

Donde utilizar Redis



Almacenamiento en caché: Crea una caché en memoria de alto rendimiento, muy utilizado como cache de páginas HTML o fragmentos de estas, lo que acelerará el acceso a las mismas, a la vez que evitamos llegar a los servidores web o de aplicaciones, reduciendo la carga en éstos



Administración de sesiones: Ideal para tareas de administración de sesiones.



Clasificaciones en tiempo real: Esto facilita la creación de clasificaciones dinámicas, realizadas directamente en memoria y desplegadas en tiempo real.

También utilizado para mantener información efímera



Cache de Base de Datos: Permite trabajar con un conjunto de datos extraídos de una base de datos, para otorgar mayor velocidad de acceso a los mismos.



Listas de elementos recientes: Es muy habitual mostrar listas en las que aparecen las últimas actualizaciones de algún elemento realizadas por los usuarios. Por ejemplo: comentarios sobre un post, últimas imágenes subidas, los artículos de un catálogo, artículos de un carrito de compras, contadores, estadísticas, etc.



Limitación de la velocidad: Redis puede medir, y cuando sea necesario, limitar la velocidad de los eventos.

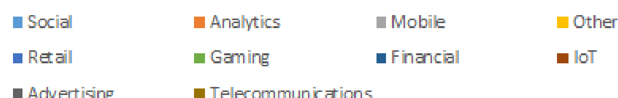


Colas: Las listas ofrecen operaciones atómicas, así como capacidades de bloqueo, por lo que resultan aptas para una variedad de aplicaciones que requieren un agente de mensajes fiable o una lista circular.



Mensajería: Compatible con el estándar PUB/SUB con la correspondencia de patrones. Apto para alto desempeño y transmisiones en tiempo real e intercomunicación en los servidores.

Tipos de aplicaciones que usan Redis



Instalación desde repositorios

Requisitos

```
apt install wget curl build-essential tcl
apt install install Redis-server
```

Instalación desde código fuente

Descarga y compilación

```
$ wget http://download.Redis.io/Redis-
stable.tar.gz

$ tar -xvzf redis-stable.tar.gz

$ cd redis-stable
```

```
$ make

$ make install
```

Configurar Redis

```
$ mkdir /etc/redis
$ mkdir /var/redis
```

Configuración del usuario que gestiona el servicio

```
$ adduser --system --group --no-create-
home redis

$ chown redis:redis /var/redis
$ chmod 770 /var/redis
```

Configuración del servicio

```
$ cp redis-stable/redis.conf /etc/redis/
```

/etc/redis/redis.conf

```
##Default port to listen reids. You can
also change it as per your need.
port  6379
## If you run Redis from upstart or
systemd.
supervised systemd
##Path of the Redis log.
logfile /var/log/redis.log
##Location of the Redis data.
dir  /var/redis/
##IP address to listen Redis on. You can
also specify multiple IP address.
bind 192.168.0.101
```

Crear el servicio Redis

/etc/systemd/system/redis.service

```
[Unit]
Description=Redis In-Memory Data Store
After=network.target

[Service]
User=Redis
Group=Redis
ExecStart=/usr/local/bin/Redis-server
/etc/Redis/Redis.conf
ExecStop=/usr/local/bin/Redis-cli
shutdown
Restart=always

[Install]
WantedBy=multi-user.target
```

Gestión del servicio

Iniciar el servicio

```
$ systemctl start Redis
```

Detener el servicio

```
$ systemctl stop Redis
```

Estado del servicio

```
$ systemctl status Redis
```

Versión instalada

```
$ Redis-server --version
Redis server v=4.0.9 sha=00000000:0
malloc=jemalloc-4.0.3 bits=64
build=6be6c3d9c9b24824
```

Acceso

```
$ redis-cli
```

Test de Redis

Prueba de respuesta

```
127.0.0.1:6379>ping
PONG
```

Asignando un valor

```
127.0.0.1:6379> set frase "Fundación
AtixLibre"
OK
```

Recuperando el valor

```
127.0.0.1:6379> get frase
"Fundación AtixLibre"
```

Acceso remoto

```
$ redis-cli –h 192.168.0.101
```

Asegurando Redis

Por defecto el acceso a Redis se la realiza sin el uso de una contraseña, en el caso de requerir mayor seguridad y control de acceso, se puede configurar a Redis a tal efecto.

Cambio de contraseña

Generar código sha256

```
echo "password" | sha256sum
a453f41f929a3297289b68788b91d1454d91c7161
3416d14020c5206cc35579e -
```

Incluir la contraseña generada en la configuración

/etc/redis/redis.conf

```
requirepass
a453f41f929a3297289b68788b91d1454d91c7161
3416d14020c5206cc35579e
```

A partir de este momento Redis solicita que estemos autenticados para poder realizar cualquier tarea

redis-cli

```
127.0.0.1:6379> ping
(error) NOAUTH Authentication required.

127.0.0.1:6379>auth password
```

Tipos de datos que maneja



Redis actualmente maneja 4 tipos de datos, descritos a continuación:

- Cadena de texto
- Listas
- Hashes o Tablas de dispersión
- Conjuntos (tanto ordenados como no ordenados)

Cadenas de texto (STRING)

La estructura más básica de Redis son las cadenas de texto, permitiendo realizar operaciones sobre ellas o en parte de las mismas, también permite incrementar/decrementar valores de enteros y reales.

Operaciones con Strings

Asignar un valor

```
<127.0.0.1:6379> set mensaje "Revista de
Software Libre Atix"
OK
```

Recuperar el valor

```
<127.0.0.1:6379> get mensaje
"Revista de Software Libre Atix"
```

Longitud del mensaje

```
<127.0.0.1:6379> strlen mensaje
(integer) 30
```

Eliminar el mensaje

```
<127.0.0.1:6379> del mensaje
(integer) 1

<127.0.0.1:6379> get mensaje
(nil)
```

Incrementar y decrementar valores

```
127.0.0.1:6379> set a "16"
OK

127.0.0.1:6379> get a
"16"

127.0.0.1:6379> incr a
(integer) 17

127.0.0.1:6379> get a
"17"

127.0.0.1:6379> incrby a 10
(integer) 27

127.0.0.1:6379> decr a
(integer) 26

127.0.0.1:6379> decrby a 5
(integer) 21

127.0.0.1:6379> get a
"21"
```

Tiempo de vida

```
127.0.0.1:6379> set mensaje2 "Mensaje Temporal"
OK
127.0.0.1:6379> get mensaje2
"Mensaje Temporal"
```

```
127.0.0.1:6379> expire mensaje2 10
(integer) 1

127.0.0.1:6379> get mensaje2
"Mensaje Temporal"

127.0.0.1:6379> ttl mensaje2
(integer) 1

127.0.0.1:6379> ttl mensaje2
(integer) -2

127.0.0.1:6379> get mensaje2
(nil)
```

Asignaciones múltiples

```
127.0.0.1:6379> mset nombre "Esteban" apellido1 "Saavedra" apellido2 "Lopez"
OK
```

Recuperar múltiples valores

```
127.0.0.1:6379> mget nombre apellido1 apellido2
1) "Esteban"
2) "Saavedra"
3) "Lopez"
```

Listas (LIST)

Permiten manejar listas enlazadas de cadenas de texto, otorgando la posibilidad de añadir o extraer items desde ambos extremos, trim en desplazamientos, leer items de forma individual o múltiple, buscar o eliminar items por su valor.

Operaciones sobre Listas

Insertar elementos a la izquierda de una lista

```
127.0.0.1:6379> lpush seriestv "Walking Dead" "Black List" "House"
(integer) 3
```

Insertar elementos a la derecha de una lista

```
127.0.0.1:6379> rpush seriestv "Revenge"
(integer) 4
```

Número de elementos de una lista

```
127.0.0.1:6379> llen seriestv
(integer) 4
```

Recuperar elementos específicos

```
127.0.0.1:6379> lindex seriestv 1
"Black List"
```

Operaciones varias

```
127.0.0.1:6379> lpush seriestv "Blindspot"
(integer) 5

127.0.0.1:6379> lindex seriestv 4
"Revenge"

127.0.0.1:6379> lindex seriestv 3
"Walking Dead"

127.0.0.1:6379> lindex seriestv 2
"Black List"

127.0.0.1:6379> lindex seriestv 1
"House"

127.0.0.1:6379> lindex seriestv 0
"Blindspot"
```

Listar rango de valores

```
127.0.0.1:6379> lrange seriestv 0 2
1) "Blindspot"
2) "House"
3) "Black List"

127.0.0.1:6379> lrange seriestv 2 4
1) "Black List"
2) "Walking Dead"
3) "Revenge"
```

Listar todos los elementos de la lista

```
127.0.0.1:6379> lrange seriestv 0 -1
1) "Blindspot"
2) "House"
3) "Black List"
4) "Walking Dead"
5) "Revenge"
```

Insertar un valor en una posición determinada.

```
127.0.0.1:6379> lset seriestv 2 "The Resident"
OK
```

```
127.0.0.1:6379> lrange seriestv 0 -1
1) "Blindspot"
2) "House"
3) "The Resident"
4) "Walking Dead"
5) "Revenge"
```

Eliminar valores de izquierda/derecha

```
127.0.0.1:6379> lpop seriestv
"Blindspot"

127.0.0.1:6379> lrange seriestv 0 -1
1) "House"
2) "The Resident"
3) "Walking Dead"
4) "Revenge"

127.0.0.1:6379> rpop seriestv
"Revenge"

127.0.0.1:6379> lrange seriestv 0 -1
1) "House"
2) "The Resident"
3) "Walking Dead"
```

Limitar número de elementos de la lista

```
<127.0.0.1:6379> ltrim listausuario 0 20
OK
```

El comando “ltrim” lo usamos para limitar los registros que tendrá la lista. Es este caso empezará en la posición cero (0) hasta la posición veinte (20).

Conjuntos (SET)

Es una colección desordenada de cadenas únicas.

Sobre esta colección se pueden realizar operaciones de adición, actualización o borrado de items individuales, listar los miembros de la colección, realizar intersecciones, uniones diferencia y actualizaciones de items aleatorios.

Operaciones sobre conjuntos

Adicionar miembros al conjunto

```
127.0.0.1:6379> sadd frameworks "Ruby on
Rails" "Django"
(integer) 2

127.0.0.1:6379> sadd frameworks "Grails"
(integer) 1
```

Listar miembros de un conjunto

```
127.0.0.1:6379> smembers frameworks
1) "Grails"
2) "Ruby on Rails"
3) "Django"
```

Verificar si un miembro existe

```
127.0.0.1:6379> sismember frameworks
"Django"
(integer) 1

127.0.0.1:6379> sismember frameworks
"Yii"
(integer) 0
```

Unión de conjuntos

```
127.0.0.1:6379> sadd padres Jose Domy
(integer) 2

127.0.0.1:6379> sadd hijos Jeanneth Jenny
Esteban
(integer) 3

127.0.0.1:6379> smembers padres
1) "Jose"
2) "Domy"

127.0.0.1:6379> smembers hijos
1) "Esteban"
2) "Jenny"
3) "Jeanneth"

127.0.0.1:6379> sunion padres hijos
1) "Jose"
2) "Domy"
3) "Esteban"
4) "Jenny"
5) "Jeanneth"
```

Diferencia e intersección

```
127.0.0.1:6379> sadd equipo_a Gabriel
David Esteban
(integer) 3

127.0.0.1:6379> sadd equipo_b Franz
Wilson Esteban
(integer) 3

127.0.0.1:6379> sadd equipo_c Gabriel
Esteban Franz Jose
(integer) 4

127.0.0.1:6379> sdiff equipo_b equipo_a
1) "Franz"
2) "Wilson"

127.0.0.1:6379> sdiff equipo_a equipo_b
1) "Gabriel"
2) "David"
```

```
127.0.0.1:6379> sinter equipo_a equipo_c
1) "Gabriel"
2) "Esteban"

127.0.0.1:6379> sinter equipo_c equipo_a
1) "Gabriel"
2) "Esteban"
```

Valores aleatorios

```
127.0.0.1:6379> srandmember equipo_c
"Gabriel"

127.0.0.1:6379> srandmember equipo_c
"Franz"

127.0.0.1:6379> srandmember equipo_c
"Esteban"
```

Eliminar elementos de un conjunto

```
127.0.0.1:6379> srem equipo_c Jose
(integer) 1

127.0.0.1:6379> smembers equipo_c
1) "Franz"
2) "Gabriel"
3) "Esteban"
```

Número de elementos de un conjunto

```
127.0.0.1:6379> scard equipo_c
(integer) 3
```

Hash

Son tablas de arrays de clave-valor. Permiten añadir, actualizar o eliminar ítems individuales y actualizaciones de la tabla

Operaciones sobre Hash

Añadir ítems al hash

```
127.0.0.1:6379> hset votacion candidato1
15
(integer) 1

127.0.0.1:6379> hset votacion candidato2
5
(integer) 1
```

Recuperar valores de un ítem específico

```
127.0.0.1:6379> hget votacion candidato1
"15"

127.0.0.1:6379> hget votacion candidato2
"5"

127.0.0.1:6379> hset votacion candidato3
13
(integer) 1

127.0.0.1:6379> hset votacion candidato4
7
(integer) 1
```

Listar los ítems del hash

```
127.0.0.1:6379> hgetall votacion
1) "candidato1"
2) "15"
3) "candidato2"
4) "5"
5) "candidato3"
6) "13"
7) "candidato4"
8) "7"
```

Incrementar valores de un ítem del hash

```
127.0.0.1:6379> hincrby votacion
candidato2 3
(integer) 8

127.0.0.1:6379> hgetall votacion
1) "candidato1"
2) "15"
3) "candidato2"
4) "8"
5) "candidato3"
6) "13"
7) "candidato4"
8) "7"
```

Decrementar valores de un ítem del hash

```
127.0.0.1:6379> hincrby votacion
candidato2 -1
(integer) 7

127.0.0.1:6379> hgetall votacion
1) "candidato1"
2) "15"
3) "candidato2"
4) "7"
5) "candidato3"
6) "13"
7) "candidato4"
8) "7"
```

Adición múltiple de ítems a un hash

```
127.0.0.1:6379> hmset votacion candidato5 2 candidato6 11
OK

127.0.0.1:6379> hgetall votacion
 1) "candidato1"
 2) "15"
 3) "candidato2"
 4) "7"
 5) "candidato3"
 6) "13"
 7) "candidato4"
 8) "7"
 9) "candidato5"
10) "2"
11) "candidato6"
12) "11"
```

Eliminar un ítem del hash

```
127.0.0.1:6379> hdel votacion candidato5
(integer) 1

127.0.0.1:6379> hgetall votacion
 1) "candidato1"
 2) "15"
 3) "candidato2"
 4) "7"
 5) "candidato3"
 6) "13"
 7) "candidato4"
 8) "7"
 9) "candidato6"
10) "11"
```

Recuperar solo llaves o valores del hash

```
127.0.0.1:6379> hkeys votacion
1) "candidato1"
2) "candidato2"
3) "candidato3"
4) "candidato4"
5) "candidato6"

127.0.0.1:6379> hvals votacion
1) "15"
2) "7"
3) "13"
4) "7"
5) "11"
```

Conjuntos ordenados (ZSET)

Son mapas ordenados de cadenas por medio de valores enteros o punto flotante y ordenados por cierto valor.

Permiten añadir, actualizar o eliminar valores individuales, actualizar ítems en base a rangos de valores o valores miembro.

Operaciones sobre conjuntos ZSET

Añadir elementos al ZSET

```
127.0.0.1:6379> zadd colores 10 rojo
(integer) 1

127.0.0.1:6379> zadd colores 2 azul
(integer) 1

127.0.0.1:6379> zadd colores 7 verde
(integer) 1

127.0.0.1:6379> zadd colores 15 morado
(integer) 1

127.0.0.1:6379> zadd colores 12 amarillo
(integer) 1
```

Listar elementos de ZSET

```
127.0.0.1:6379> zrange colores 0 -1
1) "azul"
2) "verde"
3) "rojo"
4) "amarillo"
5) "morado"
```

Listar rango de elementos del ZSET

```
127.0.0.1:6379> zrange colores 1 3
1) "verde"
2) "rojo"
3) "amarillo"
```

Listar elementos con sus valores

```
127.0.0.1:6379> zrange colores 0 -1 withscores
 1) "azul"
 2) "2"
 3) "verde"
 4) "7"
 5) "rojo"
 6) "10"
 7) "amarillo"
 8) "12"
 9) "morado"
10) "15"
```

Listar elementos en orden inverso con y sin valores

```
127.0.0.1:6379> zrevrange colores 0 -1
1) "morado"
2) "amarillo"
3) "rojo"
4) "verde"
5) "azul"
```

```
127.0.0.1:6379> zrevrange colores 0 -1
withscores
 1) "morado"
 2) "15"
 3) "amarillo"
 4) "12"
 5) "rojo"
 6) "10"
 7) "verde"
 8) "7"
 9) "azul"
10) "2"
```

Eliminar un elemento del conjunto

```
127.0.0.1:6379> zrevrange colores 0 -1
withscores
1) "morado"
2) "15"
3) "rojo"
4) "10"
5) "verde"
6) "7"
7) "azul"
8) "2"
```

Muestra el valor de un elemento

```
127.0.0.1:6379> zscore colores rojo
"10"
```

Muestra su posición del elemento

```
127.0.0.1:6379> zrank colores rojo
(integer) 2
```

Muestra su posición del elemento del conjunto en orden inverso

```
127.0.0.1:6379> zrevrank colores rojo
(integer) 1
```

Interacción Redis y Python

Redis, permite interactuar con distintos lenguajes de programación, para este caso haremos una demostración de la interacción con Python; básicamente se realizarán diferentes operaciones desde consola y se observarán los resultados desde Python y viceversa.

Instalar el cliente Redis para python

```
$ pip install Redis
```

Conexión

```
$ python
Python 3.4.3 (default, Oct 14 2015,
20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> import Redis
>>> r = Redis.Redis(host='localhost',
port=6379, db=0)
```

A continuación mostramos sólo algunos ejemplos varios del manejo de Redis desde python en sus distintos tipos de datos. Teniendo en cuenta que Python cuenta con una librería completa para el manejo de todas las funcionalidades que dispone Redis.

Manejo de Cadenas en Python

Desde consola

```
127.0.0.1:6379> set frase "Fundacion
AtixLibre"
OK

127.0.0.1:6379> get frase
"Fundacion AtixLibre"
```

Recuperar valor de la cadena desde Python

```
>>>
>>> r.get("frase")
b'Fundacion AtixLibre'
>>>
```

Asignar un valor desde Python

```
>>> r.set("mensaje", "Bienvenido a Redis
desde Python")
True
>>>
```

Recuperar el valor desde consola

```
127.0.0.1:6379> get mensaje
"Bienvenido a Redis desde Python"
```

Incrementar valores desde Python

```
>>> r.set("numero", 4)
True

>>> r.get("numero")
b'4'

>>> r.incr("numero")
5

>>> r.incr("numero",2)
7
>>>
```

Manejo de conjuntos en Python

```
>>> r.sadd("frameworks","Ruby on
Rails","Django")
2

>>> r.smembers("frameworks")
{b'Ruby on Rails', b'Django'}
>>>
```

Manejo de Listas en Python

```
>>> r.lpush("padres","Domy","Jose")
2

>>> r.lrange("padres",0,-1)
[b'Jose', b'Domy']
>>>
```

Manejo de hash en Python

```
>>> r.hmset("candidatos", {"jose":
3,"carlos":8})
True

>>> r.hgetall("candidatos")
{b'carlos': b'8', b'jose': b'3'}
>>>
```

Conclusiones

El manejo de bases de datos en memoria, han contribuido enormemente el desarrollo de aplicaciones en la web, principalmente por:

- Mejorar el rendimiento de las mismas, en cuanto a rapidez de acceso se refiere.
- Su versatilidad y facilidad de uso.

Referencias

[1] http://www.redis.org



Esteban Saavedra L.
Presidente Fundacion AtixLibre
esteban.saavedra@atixlibre.org
BOLIVIA

AtixLibre

Hacia un Futuro Innovador

Etico

Libre

Justo